ClientLight:

Aplicació web per a la gestió d'actius.

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

Juan López Villarán

i dirigit per

Yolanda Benítez Fernández

Escola d'Enginyeria

RESUM

Títol del projecte: ClientLight: Aplicació web per a la gestió d'actius.	
Autor: Juan López Villarán	**Data:** Setembre, 2013
Tutora: Yolanda Benítez Fernández	
Titulació: Enginyeria tècnica d'informàtica de Sistemes.	
Paraules clau **Català:** aplicació, web, gestió, Java, client, servidor **Castellà**: aplicación, web, gestión, Java, cliente, servidor **Anglès:** aplication, web, management, Java, client, server	
Resum del projecte **Català:** creació d'una aplicació web en servidor local juntament amb una base de dades que permetrà gestionar els actius físics d'una empresa d'enllumenat públic utilitzant el framework JSF. **Castellà:** creación de una aplicación web en servidor local junto a una base de datos que permitirá gestionar los activos físicos de una empresa de alumbrado público utilizando el framework JSF. **Anglès:** creating a web application in a local server with a database that will manage the physical assets of public lighting company using JSF framework..	

1. Introducció

1.1 Què es vol fer?

La present memòria conté tota l' informació del projecte de final de carrera i tracta sobre una aplicació web. En el context actual havent pres la web una importància tant rellevant en el terreny de la gestió empresarial, ja no només trobem portals. En el present s'aposta per aplicacions web, servidors en xarxa i anar fent el hardware de l'usuari independent de la càrrega d'execució.

Es per això que hem decidit gestar una aplicació web que giri en torn a un problema real per investigar a fons si realment és possible dur-ho a terme .

A continuació expliquem dita temàtica i com comença el projecte.

1.2 Motivació

La motivació de fer una aplicació web, a part del que s'ha explicat en el *punt 1.1* neix el 12 de setembre de 2012 quan vaig començar a fer un conveni de pràctiques amb una empresa de enllumenat públic. Degut a la meva experiència en bases de dades i programació de software de gestió adquirida a una anterior empresa, vaig ser assignat a un nou projecte com a analista de sistemes. La meva funció va ser la d'investigar, recollir els requeriments, analitzar les tecnologies més adients, dissenyar i establir el entorn de treball per que en un futur s'implementés el codi d'una aplicació web per a la gestió dels recursos de l'enllumenat que fins ara es realitzava amb una aplicació d'escriptori (programa convencional).

El marc era el de passar l'execució del programa gestor a un servidor i que els clients, interactuessin mitjançant la xarxa.

El saber fins a on es podria equiparar una aplicació accedida via web vers una aplicació instal·lada a la pròpia màquina em va cridar l'atenció fins al punt de escollir-ho com a tema per al projecte de final de carrera.

1.3 Software per a la gestió d'actius

El software per a la gestió d'actius[1] és el programari que permet dur a terme un paradigma de negoci per optimitzar les parts del cicle de vida dels actius des d'un punt de vista pro actiu (interferint). Per tant comprèn el disseny, creació, identificació/enregistrament, manteniment, prevenció d'incidents, reemplaçament i optimització de l'ús dels mateixos. A la vegada s'inclouen d'altres parts de l'empresa com el control de recursos humans, normativa i transport ja que es considera que formen part de l'estratègia en vers als actius. La distribució d'aquets relaciona diferents departaments, localitzacions o fins i tot empreses.

El grup al que pertany aquest tipus de software és l'anomena't Software enterprise (per a la gestió empresarial) i es conegut per Internet pel seu nom en anglès software Enterprise Assets Managment (EAM).

El potencial de l'estratègia és obtenir una visió integral de tota l'empresa centrada en els actius que explota per poder extreure informació, actuar en conseqüència a la vegada que poder preveure errors i millorar la productivitat minimitzant el cost total. Al treballar tota l' informació amb uns mateixos procediments es genera una sinergia entre totes les parts afectades al mateix temps que es redueix el temps de planificació i costos .

Un altre punt a destacar és l' importància de que el software sigui escalable, és a dir, que permeti futurs procediments, ampliacions i fins i tot la connectivitat amb altres sistemes. Si tenim ben definit un data wharehouse (magatzem de dades) on es emmagatzemi tot el model de negoci podrem fer-ho. Per exemple, suposem que una empresa amb EAM vol ampliar el programari per a una millor relació amb els clients i decideix implantar un CRM[2]; si es compleix el punt anterior de escalabilitat, podrem crear una vista concreta de l' informació i dissenyar interfases de cara al departament de Marketing.

[1] Wikipedia.org - És un ben tangible o intangible que posseeix una empresa o persona natural. Per extensió, es denomina també actiu al conjunt dels actius d'una empresa.
[2] Coustomer Relationship Managment - A diferència de centrar-se en els actius, és una estratègia que es centra en estructurar l' informació per potenciar les relacions amb els client. Exemple: emmagatzemament de preferències per poder fer campanyes de marketing.

1.4 Beneficis d'aquest tipus d'estratègia[3]

- Millora del 5% en la planificació, seguiment i productivitat de la mà d'obra.
- Reducció del 10% en el volum del inventari.
- Reducció del 5% en costos de manteniment de l' inventari.
- Reducció del 5% en costos per nous actius.
- Increment del 5% en la disponibilitat d'equips de treball i de material.
- Increment del 50% en l' utilització de garanties.
- Reducció del 10% en costos de materials.
- Reducció del 50% en costos de processos de compres.
- Possibilitat de realitzar estudis amb informació fiable, actualitzada e immediata de tots els components del procés.
- Anticipació de carregues de treballs, averies i manteniments imprevistos.
- Traçabilitat de tots els sistemes i procediments per una presa de decisions orientada a millorar permanentment.
- Traçabilitat de tots els elements com per exemple el preu parcial d'un actiu o de tot un procés.
- Control de les activitats subcontractades.
- Introduir un sistema empresarial suposarà un canvi a gran escala a l'empresa relacionant tant els departaments com els mateixos amb els executius.
- Com a conseqüència dels punts anteriors i degut a la naturalesa complexa del canvi, s'obtindrà un anàlisi de quin es el nivell de preparació referent a coneixements, habilitats, actituds disposició al canvi, infraestructura tecnològica i de emmagatzemament de dades.

[3] Valors basats en l'opinió del 80% dels clients que han implantat: infor EAM, IFS Applications o Oracle Assets Managemnt i que afirmen tenir aquets beneficis.

1.5 EAM vs. CMMS i ERP

Quan una empresa vol gestionar o optimitzar els seus recursos informàticament i a la vegada unificar tots els processos dins d'un mateix software té diferents possibilitats. En l'àmbit d'aquest projecte els softwares que estan mes a prop són: El CMMS[4] i l'ERP[5].

Com podem observar a la *figura.1* el software CMMS inclou els mòduls de:

- Gestió del Manteniment (Maintenance Management): relacionat amb les tasques de conservament o substitució dels recursos.
- Gestió del inventari (Inventory Management): traçabilitat del elements.
- Adquisicions (Procurement): relacions amb els proveïdors (compres, recepcions, pagaments dels productes...).

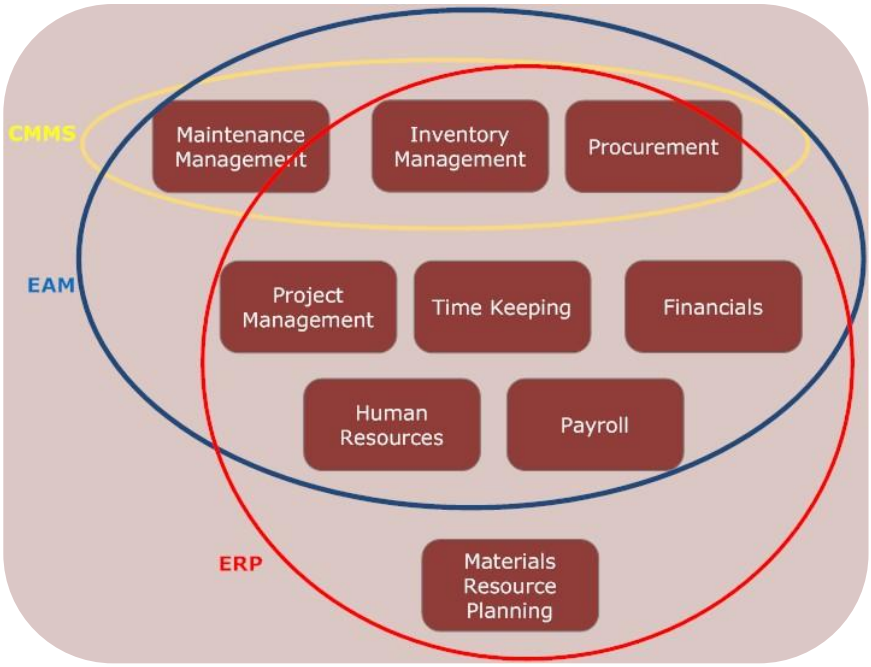


Figura 1. Mòduls comuns CMMS, ERP i EAM.

[4] Computerized maintenance management system (manteniment de recursos)
[5] Enterprise Resource Management (manteniment, finances, logística i recursos humans relacionats amb recursos)

El EAM:

- Project Management (gestió de projectes): planificació, organització, motivació i control dels recursos per aconseguir objectius específics.
- Time Keeping (control de temps): mòdul associat al control de la quantitat de temps que dediquen els treballadors a cada tasca o jornada.
- Payroll (nòmines): càlcul de quant ha de cobrar cada treballador.
- Human Resources (recursos humans): dades relacionades amb el personal o fins i tot amb candidats per a seleccions de personal.
- Financials (finances): balanços, pèrdues i beneficis o fluxos entre entitats.

El ERP:

- Materials Resource Planning (planificació dels materials): sistema de planificació i administració referent a la producció i al control d'inventari dels materials necessaris per a la seva operació. Horaris d'entrega i activitats de compra.

Així tenim que tot i que cobreixen mòduls comuns, el EAM respecte al CMMS s'incorpora quan tenim necessitat de gestionar recursos humans i finances. Amb l'ERP la diferència és l'enfocament que proporcionen els mòduls que els diferencien. Mentre l'EAM està dirigit al manteniment el ERP ho està al procés de fabricació. Per això l'entrada de compres que veiem a la *figura.2* és per orientar-la a la fabricació mentre que la entrada de la *figura.3* és per orientar-la al manteniment.

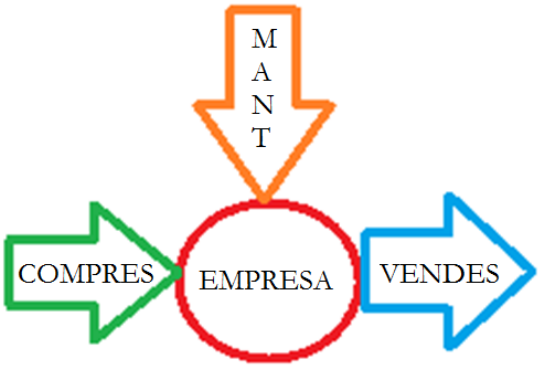


Figura 2. Entrades/sortides ERP

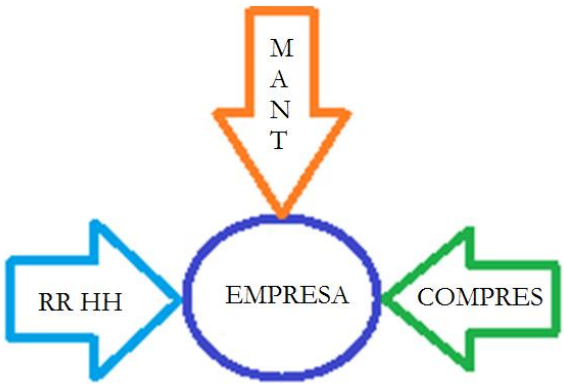


Figura 3. Entrades/sortides EAM

Finalment dir que escollir un ERP és més convenient quan els plans de producció han de ajustar-se a patrons de demanda variants així com la gestió dels materials per a la manufactura. S'ajusten a empreses amb varis productes, generalment amb plans de producció influenciats pel comportament del mercat. Els apartats bàsics que conté són: finances, manufactura, logística, vendes y recursos humans.

Escollir un EAM és convenient quan l'empresa gestiona un o pocs productes, requereix la màxima disponibilitat d'equips i els seus plans no depenen de la fluctuació del mercat. Exemples: industria minera, empreses ferroviàries, navals o com en el nostre cas una empresa que fa instal·lacions elèctriques. Els principals components són: finances, manteniment, compres/magatzems i recursos humans.

1.6 Objectius

- Establir un model d'informació que permeti la entrada, el emmagatzematge i gestió de tota l' informació relacionada amb el model de negoci sent accessible en qualsevol moment.
- L'estructura de dades ha de permetre representar qualsevol element per poder tractar-lo de manera estàndard abstractament.
- Basar l' informació en l'estat de l'actiu per poder decidir si aplicar un manteniment correctiu o predictiu.
- Dissenyar les eines necessàries per a poder dur a terme la planificació i el control del manteniment dels actius.
- Retornar l' informació en forma processada i ordenada de manera que pugui utilitzar-se per a l'avaluació de resultats i servir com a base per a la pressa de decisions.
- Tot i ser un producte genèric, s'han d'aplicar desenvolupaments específics dirigits al sector industrial i activitats que realitza l'empresa.
- Ha de permetre que puguin haver diferents interfases per satisfer les necessitats de cada departament o rol que treballi amb l'aplicació.
- Ha de existir la possibilitat de connectar i poder intercanviar dades amb altres sistemes de gestió de l'empresa com podria ser un ERP.
- Ha de facilitar l'escalabilitat per poder anar millorant i augmentant l'aplicació si així ho requereix l'empresa en el futur.

1.7 Estat de l'art

Mitjançant la web de TEC (Technology Evaluation Centers) que té com a objectiu l'anàlisi comparatiu dels softwares més rellevants per a l'empresa hem observat les millors possibilitats que hi ha al mercat. De la categoria EAM proporcionen una llista de 38 softwares. D'aquests i mitjançant una exhaustiva comparativa dels mòduls d'integració amb altres softwares, la gestió del manteniment dels actius, recursos humans, finances, compres, inventari, fiabilitat, tecnologia a la vegada que gairebé 60 submòduls dins d'aquets (ex. Dins de manteniment: gestió de les ordres de treball, manteniment preventiu, historial dels elements, manteniment de vehicles, connexió amb PDA,...).

D'aquests 8 hem seleccionat 3:

- IFS Applications, el primer és el que té la puntuació més alta a la comparativa.
- Oracle Enterprise Asset Management per que, a part de tenir la 4 millor nota, proporciona una gran quantitat de material i documents que fa que sigui el software que més fàcil seria integrar.
- Infor ja que és el que més s'ajusta als requisits del projecte.

Les característiques dels tres són semblants, softwares molt complets que incorporen totes les funcionalitats per satisfer tots els mòduls. A conseqüència d'això han de desenvolupar un tipus de software que pugui treballar sigui quin sigui el objectiu de l'empresa y el tipus de element que utilitza. Segons la nostra opinió, per grans empreses amb dinàmiques de treball molt estandarditzades són grans alternatives però pera a petites i mitjanes empreses pot esdevenir un software amb masses funcionalitats quan el que necessiten és: la representació de l' informació totalment ajustada al seu model de negoci, interfases que mostrin l' informació de la manera mes profitosa i en concordança amb l'estructura de dades i la possibilitat de escalar el software seguint els dos punts anteriors. Menys Oracle, les altres opcions no publiquen els preus i aconsellen que ells gestionin el servidor "as a Service[6]" ja que configurar en un propi servidor aquest complexos software requeririen de tècnics. No obstant es

[6] Wikipedia.org – és un model de distribució de software on les dades i el suport logístic són a servidors d'una companyia de tecnologies de l' informació i comunicació externa. S'accedeix mitjançant un client a través d'un client web.

poden instal·lar sobre les plataformes habituals (Solaris, Unix, Windows Server) si la empresa ho prefereix. Es difícil obtenir preus d'aquestes companyies per poder comparar, no obstant Oracle ho publica.

2. Recull de requeriments i estudi de viabilitat.

2.1 Introducció

En aquest capítol definirem el tipus de projecte que durem a terme des del punt de vista funcional. La part essencial es reconèixer el problema, entendre i llistar a què hem de donar resposta des del punt de vista del client. Aquesta llista s'anomenarà "recull de requeriments". Per poder analitzar en capítols posteriors el canvi de tecnologia que implantarà el software s'analitza la situació actual.

Un cop definides les funcions del software exposarem les alternatives, justificarem la opció escollida i estudiarem la seva viabilitat. Per això tindrem en compte els requisits legals i estàndards que hem de complir i les restriccions del sistema.

Al obtenir una llista de requisits a implementar, una opció escollida per fer-ho essent la mes apropiada per al cas i una justificació de la viabilitat tant econòmica com tècnica com legal podrem passar al següent capítol per planificar temporalment el projecte.

2.2 Definició projecte

El projecte d'implantació d'un software basat en l'estratègia EAM pel control i gestió del cicle de vida d'element d'una companyia de construccions elèctriques. El sistema permetrà la unificació de tot el model de negoci gracies a l'establiment d'una estructura de dades i una manera de gestionar la informació capaç de fer-ho. D'aquesta manera per exemple és el mateix de cara al sistema l' informació d'un empleat que d'una bombeta ja que estaran a una base de dades comú representats de la mateixa manera.

Allotjarem una sèrie de recursos software a un servidor ja existent de l'empresa. Establirem un servei web mitjançant un servidor web que oferirà mitjançant un navegador, una aplicació web.

És una mena de software as a Service però que no l'ofereix una companyia externa si no des de la central de la mateixa empresa. En aquest cas per l'estructura de l'empresa (una central amb servidor i delegacions client) és més adient aquest desplegament.

En cas de tenir un altre tipus d'empresa s'hauria d'analitzar si és més rentable col·locar un servidor o contractar un servidor as a Service i posar el software en el mateix.

Com es pot observar a la *figura 4*. a partir d'una zona geogràfica el tractament de l' informació es basa en delimitar diferents àrees o punts pel seu interès com a elements del negoci. Són aquests:

- Delegació: representa l'àrea que compren un municipi o ciutat. Exemple delegació de Sabadell.
- Contracte: representa l'àrea d'un barri o urbanització.
- Zona: depenent de l'extensió del contracte es subdivideix en varies zones (línea vermella.
- Punt de llum: cadascun dels fanals o sistema d'iluminació a una coordenada determinada.
- Línia: conjunt de punts de llum (sol ser un carrer).
- Punt de manteniment: node que alimenta a una o varies línees.
- Bombeta: element que genera la llum.
- Llumenera: element on es col·loca la bombeta.
- Suport: element d'ancoratge de una o varies llumeneres.

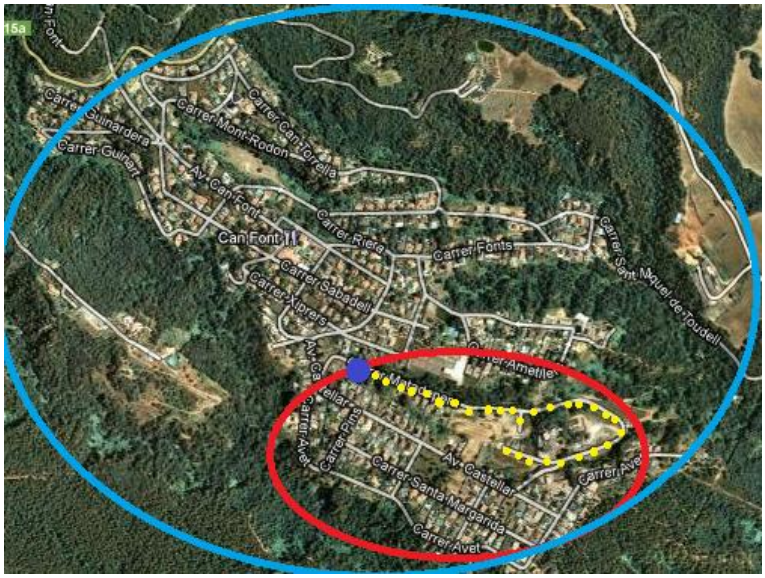


Figura 4. Elements del negoci.

A més a més hi ha d'altres recursos que considerarem al software però que són recursos humans. Tenim:

- Empleat intern: persona en nomina de l'empresa.
- Empresa externa: que proveeix a l'empresa serveis, material o personal.
- Empleat extern: persona que treballa des de una altre empresa com a subcontractat.

2.3 Definició funcionalitat per mòduls

En la primera estació en l'instal·lació d'un software empresarial gestor d'actius s'implanten una sèrie de funcionalitats estàndard que són comuns a tots els projectes. A partir d'aquí, depenent de l'empresa, pot ser que tinguin mòduls per altres funcionalitats o que els facin exclusivament sota requeriments precisos del client. En el nostre cas treballarem sobre els bàsics per garantir que establim una bona infraestructura que en el futur ens permeti una escalabilitat sòlida.

A l'hora d'escollir què considerem com a funcionalitats bàsiques ens hem basat en la web de Technology Evaluation Center. En aquesta web trobem una comparativa molt precisa i extensa però que a l'hora inclou un quadre resum dels blocs principals i les seves funcionalitats (que són les que hem utilitzat):

- General CMMS (Computerized Manteniment Management System): dins del context del EAM considerem aquesta part com una mena de taulell on es mostren dades com a recordatoris, manteniments preventius o avisos d'incidència. Inclourem:

    - Alertes/indicadors de espai en base de dades, funcionament general.
    - Calendari scheduler amb recordatori de les tasques.

- Financials: en aquesta part les dades han de ser de caire més visual ja que es tracten temes econòmics. Inclourem:

    - Outputs, elements normals (com ara botons) per interactuar.
    - Charts (gràfics interactius).

- Manteniment: en aquest cas hem considerat que una manera de fer el manteniment efectiva serà el incrustar un suport geogràfic mitjançant GMaps.

- Compra : en aquest mòdul mostrarem les dades mitjançant grids interactives i disposarem una funcionalitat de conversió de les mateixes a arxius i que es puguin descarregar.

- RRHH: contindrà elements com els del bloc de compra però aquest s'ha de poder comunicar amb les incidències que s'introdueixen al scheduler del CMMS o Manteniment.

2.4 Definició usuaris

Els usuaris o actors de l'aplicació són:

- Administrador: de caire tècnic, té rols a totes les funcionalitats de l'aplicació. Té la responsabilitat de que funcioni l'aplicació, les còpies de seguretat, el servidor, accessos i privilegis.
- EmpleatCompres: té accés al mòdul Board, Compra i Manteniment. Té la responsabilitat de gestionar l'inventari, les compres i les garanties.

- EmpleatManteniment: Board i Manteniment. Té la responsabilitat de recollir incidències i programar els manteniments.
- EmpleatRecursosHumans: Board, Manteniment RRHH. Té la responsabilitat d'assignar el personal i gestionar les actes de treball.
- Responsable: és l'encarregat de gestionar la part financera i només veu la pestanya Financials.

2.5 Estudi de la situació actual

Lògica del sistema:

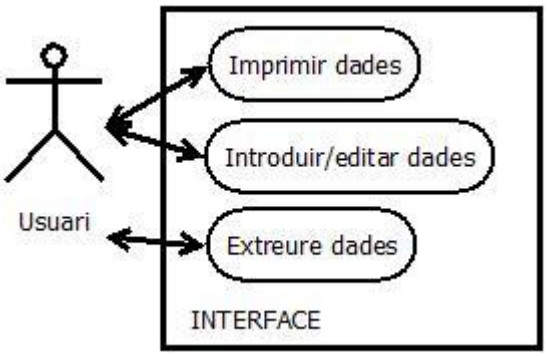


Figura 5. Lògica actual del sistema

Descripció física:

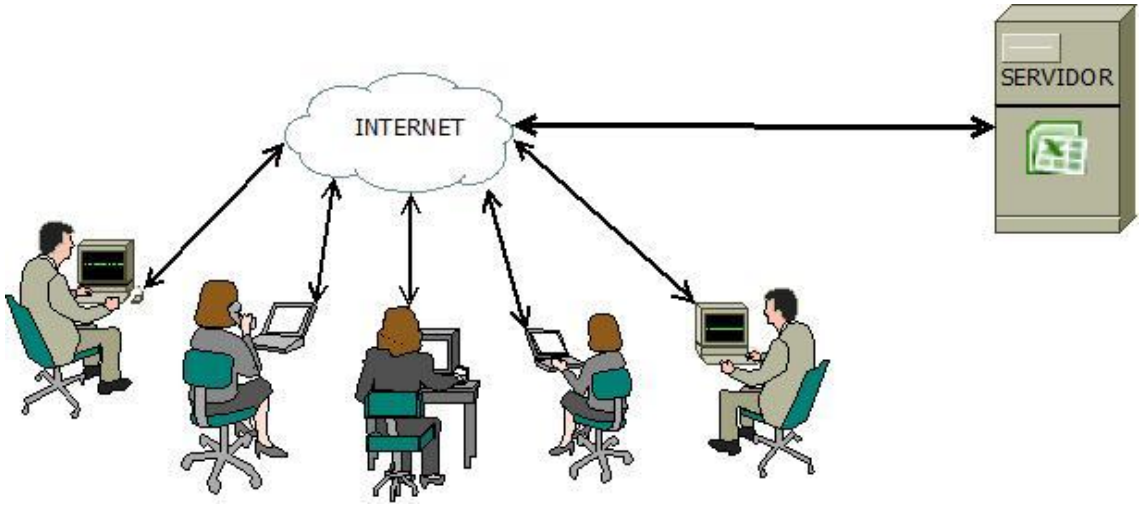


Figura 6. Descripció física actual del sistema

- El diagnòstic analitzant la situació actual és que es disposa d'un entorn arcaic, és insegur i no aporta beneficis a les tasques realitzades pels treballadors.

2.6 Requisits funcionals

- El sistema controlarà l'accés a l'aplicació dels usuaris comprovant les credencials a la base de dades.
- En cas de ser correctes les credencials mirarà quins permisos té l'usuari que accedeix i ocultarà les parts a les que no té permisos.
- A la pestanya d'inici apareixerà un calendari amb les tasques del mes
- Els mòduls més rellevants també apareixeran a aquesta pestanya
- Al clicar a la pestanya de financials es veurà el capital inicial, final i un gràfic que els relacioni.
- Es detallaran els costos des d'un punt de vista d'actiu, conjunt (línea de punt de manteniment, punt de manteniment) procés o global.
- El gràfic tindrà links als punts de màximes pèrdues.
- La pestanya manteniment contindrà un cercador d'elements per tipus que llistarà tots els d'un tipus seleccionat.
- Al clicar sobre un d'aquets apareixerà tota l' informació relacionada al element.
- A part de l' informació alfanumèrica un google maps indicarà la situació del element.
- Si des de el mapa es clica sobre un element ha d'aparèixer l' informació alfanumèrica.
- Des de aquesta pestanya tindrem accessible un botó anomenat "programar manteniment".
- Programar manteniment relacionarà mitjançant buscadors els elements, les parts afectades i si és de tipus preventiu, correctiu o predictiu.
- En la pestanya de compres hi haurà un cercador de productes per tipus que carregarà un altre per element i un tercer per quantitat.
- Amb els tres cercadors omplerts al clicar a un botó sortirà el preu amb IVA a un quadre de text.
- Els productes s'aniran carregant a una graella al costat.
- Al acabar la operació apareixerà una factura.
- Es pot sol·licitar imprimir o passar a PDF.

- A la pestanya de RRHH apareixerà la llista de tasques pendents ordenades per prioritats.
- Aquí estarà la funcionalitat d'assignar equips de treball a les tasques.
- Qualsevol treballador mitjançant doble clic podem obtenir la seva informació.
- Al prémer el botó sortir o al sortir directament de l'aplicació tancant el navegador les dades han de quedar emmagatzemades i la sessió del usuari ha de quedar tancada.

2.7 Requisits no funcionals

- Degut a que l'empresa ja té una base de dades Oracle instal·lada amb la que ha estat funcionant molt de temps el software haurà de muntar sobre la mateixa.
- S'ha de fer un estudi de la base de dades i en el cas de que tingui una bona estructura s'ha d'aprofitar integra, en cas contrari s'haurà de migrar a una nova estructura però mantenint les dades.
- El temps de comprovació a l'entrada de l'aplicació ha de ser de menys de 5 segons.
- La aplicació ha de permetre l'entrada concurrent de varis treballadors.
- S'ha de fer un test per assegurar que el màxim de clients que entraran a l'aplicació pot fer-ho sense penjar-la i afegir un 30% de marge.
- Tots els arxius que es pugin han de ser en format PDF.
- El servidor que té l'empresa contindrà l'aplicació web i la base de dades.
- S'instal·larà un servidor d'aplicacions JBoss al servidor.
- Al servidor mitjançant tecnologia JAVA Web (J2EE) es muntarà l'aplicació.
- Cada delegació (clients de l'aplicació) tindrà una URL d'accés al servidor i un Servlet individual gestionarà la seva zona.
- La tecnologia ha de ser compatible amb Android.

2.8 Restriccions del sistema

- Degut a que l'empresa té d'una altra aplicació un servei de base de dades amb el motor Oracle instal·lat, s'ha d'aprofitar.
- La connexió ha de ser amb jdbc perquè en el futur volen connectar sota aquesta tecnologia altres aplicacions.
- En general s'han de fer proves amb el software/hardware existent per comprovar que tot és compatible.
- S'han d'establir entorns de desenvolupament, proves, pre-producció i producció.

2.9 Normatives, estàndards i legislació

LOPD (Ley Organica de protección de datos). Llei referent a la privacitat de les dades per garantir el dret al honor, intimitat personal i familiar i la pròpia imatge, així com les tasques relacionades per que això es compleixi i no hi hagi cap pèrdua d'informació.

- Les dades personals han de ser reals i s'ha d'assegurar que romanen en secret.
- Al recollir dades mitjançant formularis s'ha d'expressar que s'aplicaran les mesures d'aquesta llei i s'ha de requerir un consentiment explícit.
- Ha d'haver la possibilitat d'exercir els drets d'accés, rectificació i cancel·lació.
- No es poden emetre dades a terceres parts.
- S'ha de gestionar el accés a les dades.
- Obligació de emmagatzemament de backups setmanals.
- S'han de realitzar proves sense dades reals.

ISO 9000, 20000. Estàndards de qualitat que estableix un mapa de processos per cobrir les necessitats del negoci al que s'ofereix software, entrega dels recursos acordats i minimitzant els riscos d'overload (sobrecàrrega).

- Definició correcta dels serveis oferts respecte a les necessitats del client.

- Procediments per impedir interrupcions o per recuperació en el mínim temps possible.
- Control dels canvis en la configuració, incidents i versions així com el software i hardware instal·lat en el entorn.
- Millora permanent dels serveis existents tenint en compte els errors passats.
- Gestió de la seguretat de l' informació.
- Generació d'informes del servei.

PAS-55. Estàndard de qualitat equivalent als ISO mencionats però orientat a la gestió d'actius:

- Planificació: s'ha de proporcionar una estratègia pel cicle de vida dels actius, definir els objectius dels mateixos i definir els plans de contingència en cas d'error.
- Actuació : s'ha d'establir l'estructura, autoritats i responsabilitats així com el control d'activitats per tercers, els riscos i els processos d'informe de tot plegat.
- Verificació : processos per assegurar que s'ha implementat correctament o quins errors hi ha hagut i deixar constància.
- Revisió: monitorització dels errors, no conformitats i èxits, accions de millora.... per tal de analitzar i extreure millores permanents.

2.10 Alternatives i viabilitat econòmica de la solució proposada

El que primer hem de saber és quin estalvi tindrem un cop instal·lat el software per poder saber en quan de temps l'amortitzarem. Per afinar aquest resultats s'hauria o bé, de provar les opcions per separat i analitzar el resultat o fer una simulació amb dades reals passades. Ambdues estan fora de l'abast d'aquest projecte i de qualsevol real. Per això es fa una aproximació aprofitant els beneficis del *apartat 1.3* de fent un càlcul predictiu.

Degut a que no es una empresa de nova creació sabem que hi ha:

- 3 treballadors: 1 encarregat de la gestió del inventari i compres, 1 en la captació d'incidències i previsions de manteniment i 1 en l'assignació de personal a les tasques. En total treballen 120 hores setmanals.

  2 equips de dues persones/equip fixes que fan funcions de: classificació inventari, revisió dades energètiques, revisions visuals, canvi bombetes, altres reparacions. En total treballen 160 hores setmanals.
- 2 equips variables depenent de la feina. En total treballen 40 hores setmanals.

Al instal·lar els mòduls de Financials, Manteniment, Compra i RRHH els beneficis que obtindrem amb més probabilitat[7]:

1. 5% planificació
2. + 5 % disponibilitat equips de treball.
3. - 5% costos nous actius degut al aprofitament de garanties i volum d'inventari

	Gestió inventari	Compres garantia	Recollir incidències	Assignar personal	Actes treball	Programació manteniment	Hores planif.	Hores equip
Empleat 1							*	*
Empleat 2							40	
Empleat 3							40	
Equips fixos								160
Equip variable								40

* Les tasques del empleat 1 no es beneficien dels punts 1 i 2 de millora de planificació i disponibilitat però si que el retorn d'un 50% més de garanties, els costos de manteniment del inventari i la reducció de 5% sobre la compra de nous actius.

Degut a que l'empresa té un software de gestió de manteniment descartem el 10% en la millora de la gestió del inventari.

[7] Són els beneficis que sempre s'obtenen al implantar aquest tipus de sol·lució. Els altres s'obtenen però amb menys probabilitat. Al no posar-los deixem un factor de seguretat i ens ajustem més realísticament.

Costos:

80 hores planificació setmanals a 10€ → 38400€

200 hores treball equips setmanals a 10€ → 96000€

Els costos dels actius anuals ascendeixen a 16800€:

- 200 bombetes x 35€
- 12 llumeneres x 400€
- 5000€ altres (cables, connectors, aparells de mesura, eines,...)

Quadre estalvi:

	Cost sense l'estratègia	Estalvi amb l'estratègia
1. -5% costos planificació	38400€	1920€
2. +5% disponibilitat equips	96000€	4800€
3. -5% costos nous actius	16800€	840€
	TOTAL ESTALVI	7560€

Implantació i personalització de la solució: Oracle Enterprise assets Management.

Cost:

1 setmana d'anàlisi i disseny:

- 40h d'analista a 20€/h → 800€
- 20h de director de projecte a 30€/h → 600€

1setmana de personalització i contractació del servei:

- 40h de programador a 20€/h → 800€
- 20h de director de projecte a 30€/h → 600€

Llicència : 4500€

Preu del servei bàsic per 3 anys: 3000€

TOTAL: 10300€

Solució proposada. Implantació d'un EAM on-premise personalitzat JAVAWeb

Pressupost:

1 setmana d'anàlisi i disseny:

- 80h d'analista a 20€/h → 1600€
- 40h de director de projecte a 30€/h → 1200€

codificació i personalització del servei:

- 120h de programador a 20€/h →2400€
- 40h de director de projecte a 30€/h →1200€

Proves i implantació

- 80h tester a 15€/h → 1200

Formació

- 15 h formació a 55€/h → 825

Upgrades 3 anys 2000€

TOTAL: 10425€

AMORTITZACIÓ EN 18 MESOS

3. Planificació del projecte

3.1 Introducció

Seguidament proposem la planificació del projecte. En aquesta part tractem de gestionar el temps i recursos disponibles. Dividim el projecte per fases i assignem ,mitjançant la metodologia de gestió de projectes, el temps necessari. Per altra banda preveiem els riscos, la seva perillositat i les pèrdues que ens poden ocasionar.

3.2 Fases del projecte

El projecte segueix un conjunt de fases amb l'objectiu de satisfer totes les parts referents al mateix (documentació, implantació, manteniment,...). A la *figura 7.* podem observar de esquerra a dreta un diagrama WBS (diagrama jeràrquic en forma de arbre) on s'il·lustren.

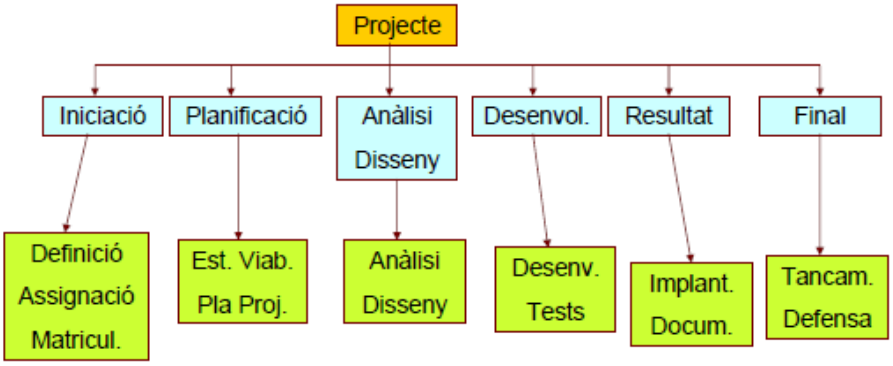


Figura 7. Diagrama WBS

Tot i que les fases de cara al projecte s'han anat presentant de manera Waterfall (es a dir, fins que no es termina una tasca no es comença l'altre), el treball personal ha sigut més de l'estil RUP (*figura 8.*). Això es degut a que durant les primeres fases on esculls el tema, has de fer algunes proves de que l'entorn funcionarà, l' implementació serà possible així com que certs recursos (software, compilador, frameworks, bbdd, servidor de proves,....) els tindrem a l'abast. De la mateixa manera al estar a la fase de implementació s'han hagut de fer canvis en l'apartat de disseny.

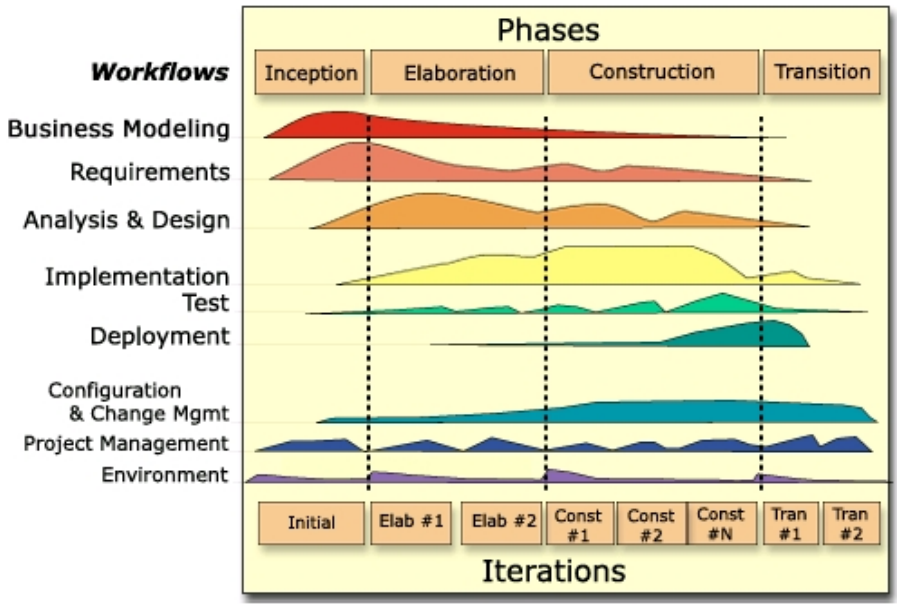


Figura 8. Fases de la metodologia RUP

En cas de portar aquest projecte a una empresa real el més adient seria utilitzar metodologies de gestió de projecte àgils. Això està contemplat, als capítols següents per tal de:

- Poder oferir software en poc temps per treballar sobre aquest i fer entregues fluides.
- Dissenyar de manera que els requisits puguin ser tornats a interpretar o canviar.
- Treballar sobre una estructura que entengui els responsables del negoci.
- En general l'atenció continua per millorar el sistema desenvolupant conjuntament amb els interessats.

Mitjançant reunions amb la professora de projecte s'han establert les dates de control i entrega de les diferents tasques del projecte.

Outlines dates:

A. Iniciació, context i abast del projecte: 18/12/12
B. Recull de requeriments i estudi de viabilitat: 28/01/13
C. Planificació i tancament de la definició: 11/03/13
D. Anàlisi i disseny: 01/04/13
E. Codificació i implementació: 21/05/13
F. Implantació, tests i manteniments: 17/06/13
G. Documentació: 25/06/13

3.3 Planificació temporal d'activitats

A continuació estudiarem la dedicació temporal de cadascuna de les activitats tant en dies com en càrrega d'hores mitjançant la següent taula:

	Iniciació	Requeriments	Planificació	Disseny	Implementació	Tests	Documentació
Dies	24	30	29	16	36	19	6
Hores	15	20	15	30	180	38	20

Com s'ha pogut deduir hi ha fases en les que es treballen més hores per dia i per tant la dedicació es major. Per veure-ho de manera visual a continuació en la *figura 9.* podem veure com l' implementació, test i disseny reben la major càrrega mentre que l'inciació, requeriments, planificació i documentació en reben menys.

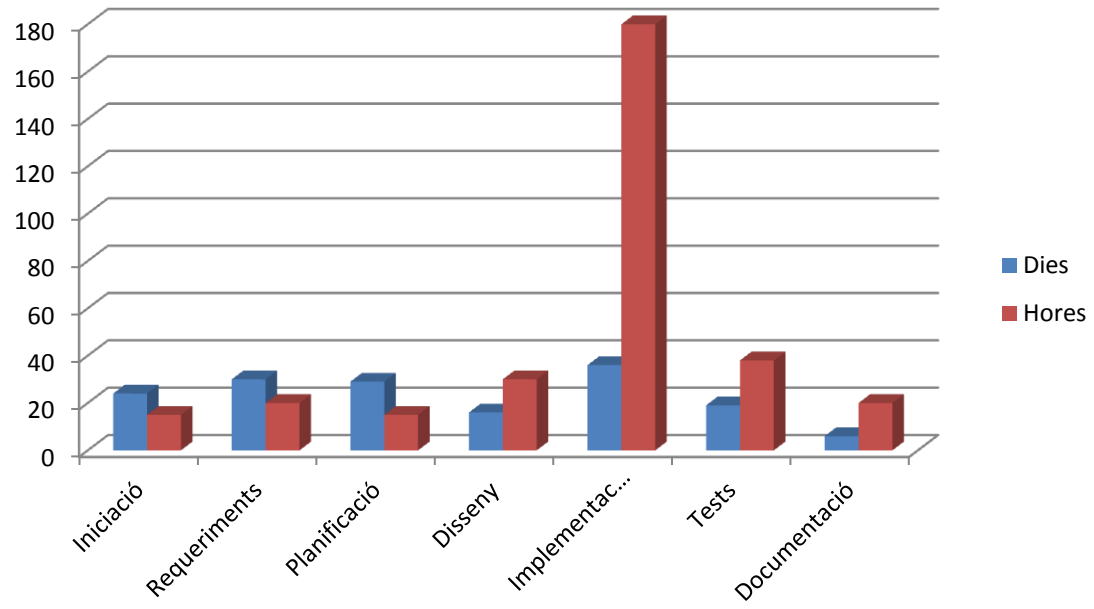


Figura 9. Comparació de dedicació horària entre fases

Un cop esmentada la dedicació, durant el transcurs d'aquest projecte, en cada fase s'ha establert una data d'entrega (temps estimat) i un temps optimista/pessimista com es pot veure a la taula següent:

Activitat	Precedent	T.optimista	T.Estimat	T.Pesimista	V	σ
A		11	15	16	0,64	0,8
B	A	18	20	23	0,64	0,8
C	B	13	15	17	0,49	0,7
D	C	20	30	39	10,24	3,2
E	D	155	180	205	68,89	8,3
F	E	28	38	48	10,89	3,3
G	F	18	20	22	0,49	0,7

El temps estimat correspon a la formula:

$$t_e = \frac{t_a + 4t_m + t_b}{6}$$

Els dos últims temps han sigut calculats aproximant i afegint uns dies més o menys al temps estimat depenent la tasca a realitzar.

Les dues últimes columnes corresponen a les formules:

$$\sigma^2 = \left(\frac{t_b - t_a}{6}\right)^2 \qquad \sigma = \frac{t_b - t_a}{6}$$

respectivament. Indiquen un rati de variació respecte la mitjana del temps. Les utilitzarem per calcular probabilitats a continuació.

En la *figura. 10* presentem el graf de Pert de les activitats del projecte. Per interpretar-ho hem de considerar que el centre indica la activitat seguint la nomenclatura del apartat 3.2 a la llista de outlines. A més a més:

- El valor numèric superior esquerra és el l'hora en que s'ha de començar la tasca.
- El valor numèric superior dret el temps com a màxim en que s'ha d'acabar la tasca.
- El valor de sota central la duració en hores que té.

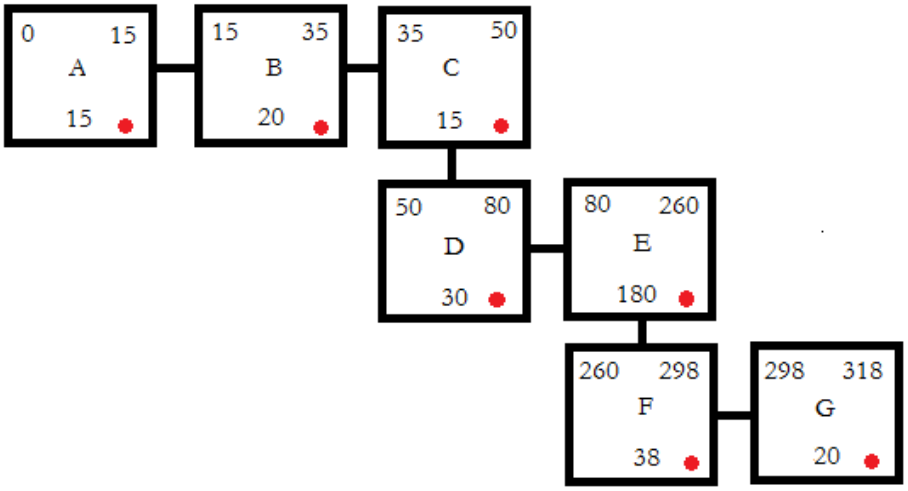


Figura 10. Graf de Part

A partir del graf anterior observem:

- Totes les tasques depenen de la finalització de la seva predecesora. Tot i que a la pràctica ha sigut tal com s'explica en l'apartat 3.2 (de manera RUP) de cara a l'elaboració i entrega de les parts s'ha considerat així.
- En conseqüència en el graf no hi ha una data mínima/màxima de començar/acabar la tasca (no hi ha folgança). Si una tasca s'ha endarrerit ha afectat a la següent.
- El camí crític (●) és tot el projecte (A,B,C,D,E,F,G). Això vol dir que en cas que qualsevol tasca no es realitzi a temps es veurà afectat tot el projecte quedant sense finalitzar alguna part.

Aprofitant la síntesi anterior donem resposta a una predicció probabilística:

a) Ens agradaria afegir una part de interconnexió amb Android i necessitaríem uns 10 dies. Si el temps estimat es de 318, quina probabilitat hi ha d'acabar en 308 per poder fer dita part?

$$z = \frac{318 - 308}{17{,}8} = 0{,}585$$

$$z = \frac{t.estimat - t.necessari}{\sum desviacions}$$

$$\frac{z2 - z1}{p2 - p1} = \frac{z2 - z}{p2 - p}$$

$$\frac{-0{,}6 - (-0{,}4)}{0{,}274 - 0{,}345} = \frac{-0{,}6 - 0{,}585}{0{,}274 - p}$$

$$p = 69\%$$

3.4 Recursos del projecte

El projecte es desenvoluparà en un servidor local a una màquina:

ASUS A53S:

- Processador Intel Core I7 2.2GHz
- 8GB memòria RAM DIMM3
- 320Gb disc dur SATA reservats
- Windows 7 x64

Es descriu a grans trets el material utilitzat per fase:

- Iniciació: analista i director de projecte. Material ofimàtic, Internet i recursos bibliogràfics.
- Requeriments i estudi de viabilitat: analista, programador i director de projecte. Material ofimàtic, programes de disseny i UML (Dia).
- Anàlisi i Disseny: analista, programador i director de projecte. Material ofimàtic, programes de disseny i UML (Dia i Eclipse Indigo amb framework eUML2).
- Codificació I implementació: analista, programador i director de projecte. Material ofimàtic, programes de disseny i UML, motor de base de dades, compilador, debugador i altres eines de desenvolupament com llibreries JBoss.
- Implantació, test I manteniment: JUnit.
- Generació de la documentació: material ofimàtic.

Un cop finalitzat el projecte i aprofitant el servidor de la central, instal·laríem al mateix i desplegaríem tenint en compte que disposaríem de:

Servidor DELL PowerEdge R610 Rack Server amb:

- Intel Xeon 5500 2.13GHz
- 64GB DDR
- 3TB SATA
- Microsoft Windows Server 2008 x64

De cara a l' instal·lació a les maquines clients de l'empresa aprofitem totes les màquines revisant que :

- Disposen de les actualitzacions de Microsoft Officce 2007.
- Google Chrome o Mozilla Firefox amb els plugins actualitzats.
- Adobe Acrobat Reader PDF actualitzat.

Software reutilitzable, persones, eines sw/hw.

3.5 Calendari del projecte

Al calendari següent es mostren les tasques en negre, sub-tasques en blau i les entregues amb un rombe negre. A més a més les fletxes indiquen la dependència entre tasques, es a dir, la tasca que precedeix la fletxa ha d'estar finalitzada per començar a la que apunta.

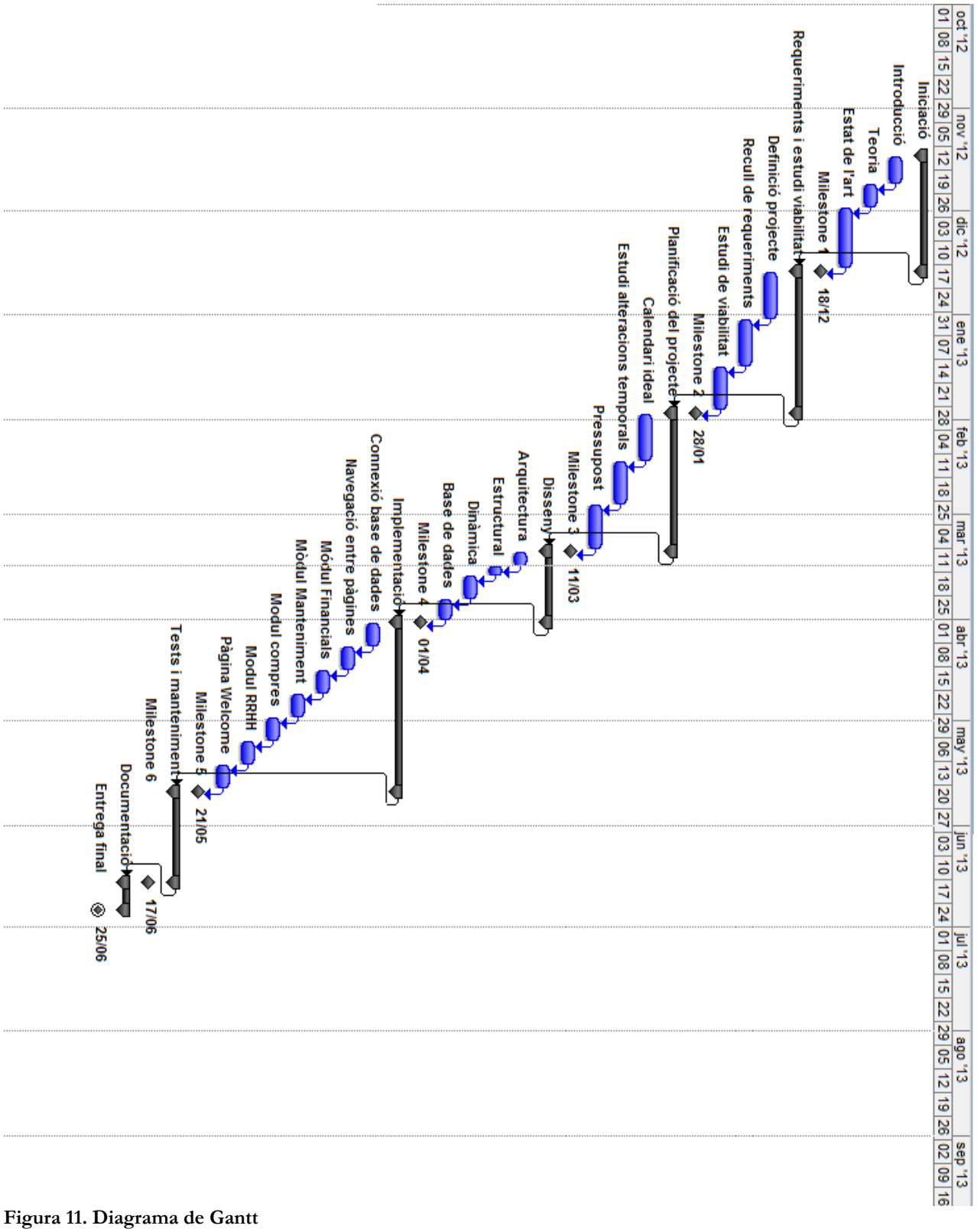


Figura 11. Diagrama de Gantt

3.6 Avaluació riscos

Ordenats per la fase on podria esdevenir el risc, problema o error, llistem:

- R1. (Requeriments/viabilitat) Falta de comunicació o precisió a l'hora de recollir els requisits.
- R2. (Requeriments/viabilitat) Dificultat per accedir als stakeholders: recollida poc precisa dels requisits.
- R3. (Planificació projecte) Planificació temporal poc acurada: no es pot acabar a la data prevista i en cas de ser un projecte implementat a una empresa augmentarien els costos en conceptes de hores de treball o bé assumir nosaltres els costos.
- R4. (Planificació projecte) Pressupost poc ajustat: degut a no haver planificat acuradament la planificació temporal es pot desquadrar el pressupost. En aquest cas augmentarien els costos del client, s'hauria de retallar alguna funcionalitat o assumir nosaltres els costos.
- R5. (Disseny) Interpretació errònia de les estructures de disseny o els fluxos entre elles. S'hauria d'analitzar i dissenyar,tornar a revisar requisits comportant un augment de temps (cost o pèrdues per la nostra part)
- R6. (Disseny/Implementació) Canvi de requisits: si són dintre de l'estructura bàsica que veurem als capítols de disseny, els canvis de requisits poden modificar les dates d'entrega i finalització del projecte.
- R7. (Disseny/Implementació) Canvi de l'estructura del projecte: reformulació general del mateix.
- R8. (Implementació) Impossibilitat de dur a terme qualsevol part del software per manca de coneixement. Pèrdua de funcionalitat o fins i tot abandonament del projecte. Pèrdua total dels beneficis.
- R9. (Proves) Falla algun test o no es detecta algun bug. Possibilitat de pèrdua de dades incomplint alguna norma, legislació o estàndard. Pèrdua econòmica i repercussions legals.
- R10. Abandonament del projecte abans de finalitzar: pèrdues econòmiques.

Tot seguit, en la taula següent la valoració de la probabilitat que succeeixin els riscos i el impacte que això causaria:

RISC	Probabilitat	Impacte	Pla de contingència
R1	Mitjana	Lleu	Tornar a entrevistar als stakeholders. Revisar els requisits.
R2	Baixa	Crític	Comunicar als responsables dels stakeholders i a ells mateixos la necessitat de participar per obtenir un resultat acurat.
R3	Mitjana	Lleu	Ajornar l'entrega final o posar més personal per entregar a temps. Assumir els costos si és per que hem estimat malament.
R4	Baixa	Lleu	Estudiar si ha sigut una mala estimació nostra i assumir els costos.
R5	Baixa	Crític	Estudiar les causes, ajustar les dates o suprimir funcionalitats per falta de temps.
R6	Mitjana	Lleu	Canvis lleus en les dates d'entrega.
R7	Baixa	Fatal	Tornar a formular el que sigui necessari tenint en compte que s'encareix el projecte. Fer un nou estudi de viabilitat tenint en compte els fets.
R8	Baixa	Crític	Buscar alternativa amb les mateixes eines o proveir de noves compatibles amb el software.
R9	Baixa	Fatal	Si es detecta anomalia, posar en quarantena el sistema i estudiar la traçabilitat per estimar que s'ha perdut.
R10	Baixa	Fatal	No hi ha solució.

3.7 Pressupost final

Si bé al *apartat 2.10* mostràvem quant costaria l' implementació de la solució en un entorn real, en aquest darrer capítol hem tractat la planificació com a projecte docent. Degut a això els costos són:

- Portàtil per desenvolupar Asus A53S → 650€
- Eclipse, frameworks i plugins → 0€

- Oracle Database → 0€
- Llibreria Primefaces → 0€
- Material de backup (disc dur) → 50€
- Impressió documentació i CD → 22€

TOTAL 722€

4. Disseny

4.1 Introducció

Un cop introduït i definit el projecte, recollits els seus requeriments i estudiada la viabilitat del mateix en el present capítol establirem el disseny. En un projecte real, si hem arribat a aquest punt, vol dir que:

- Hem realitzat un estudi de "per a qui" i "què" projecte farem.
- S'ha determinat si el mateix pot ser viable.
- Hem accedit a recollir requeriments i establir més acuradament l'abast, parts interessades, recursos, requisits i costos.
- El client ha rebut en format document els punts anteriors i ha signat la realització del projecte amb nosaltres.

Degut a que estem utilitzant un format de memòria de projecte docent, al disseny no resumirem els capítols anteriors com es fa quan s'entrega el capítol en un arxiu independent.

Aquesta part és un informe que serveix com a guia de la codificació i implementació dels sistemes i aplicacions necessàries per realitzar projecte. Per això descriu en detall el disseny de l'arquitectura i tots els components que la conformen. També és necessària per a les proves que s'han de fer ja que el resultat de les mateixes han de coincidir amb el comportament que aquí es detalla.

4.2 Arquitectura MVC

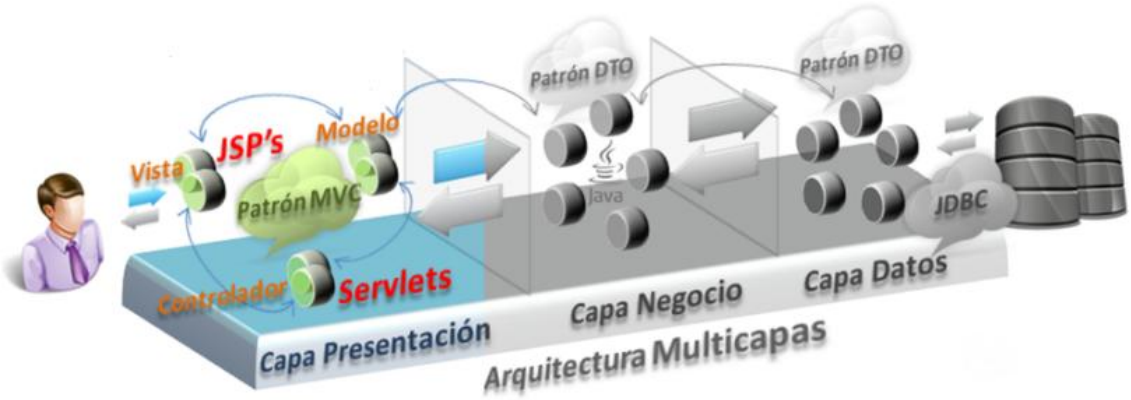


Figura 12. Arquitectura MVC

Anomenem arquitectura MVC la que està composada per:

- Capa de Presentació, vista o interfase: on es mostren els components que interactuen amb el client.
- Capa de negoci o lògica: on es controlen els estats, events i es duen a terme les operacions lògiques.
- Capa de model de dades i persistència a la base de dades: extracció i inserció de dades amb les estructures que es defineixen.

4.3 ClientLight, orientació a producte

A part de la funció que té el disseny que s'ha esmentat al *apartat 4.1*, cal aclarir que ,a part de servir de guia per a traduir els requeriments a funcionalitats, volem aportar una continuïtat a la nostra tasca establint una línea de treball que persegueix la modularitat. Així els components i la seva implementació estan orientats per a poder ser reutilitzats en altres projectes o fins i tot poder crear petits productes que es puguin afegir com si fossin plugins. Dita reflexió comporta la decisió d'unes pautes de programació que determinen com d'independents són els elements, es a dir, quina abstracció tenen així com les parts que necessitaran tenir les aplicacions que els continguin.

La paraula ClientLigth està format per client i light. La part client prové de la idea client-servidor de programació web i light fa referència a la línea de treball cap a la lleugeresa. Mirant-ho des de la perspectiva del branding (procés de construir una marca), el nom pretén reflexar que el client que ho implanti, tindrà una interfase d'usuari lleugera, ràpida i que no s'haurà de preocupar de instal·lacions ni capacitat per poder executar-ho.

Com a software empresarial ens interessa que la nostra aplicació sigui suficientment adaptada a cada usuari com per a que li proporcioni valor afegit al seu treball així com suficientment modular com per a que ens resulti rentable vers el temps dedicat.

Primerament, sabem que l'aplicació web es composa d'una part servidor i una altre client. La part servidor es "albergada" per el que anomenem servidor

d'aplicacions (contenidor). Si la part a la que desenvolupem ja té un servidor cap la possibilitat de que vulgui aprofitar-ho i ho estableixi com a requisit. Deduïm que el nostre projecte ha de ser el més independent possible a quin sigui el servidor.

El client utilitzarà un navegador per a accedir a l'app (aplicació). Aquí el que hem de fer es treballar amb tecnologia estàndard que sigui acceptada i funcioni correctament als navegadors més usats.

De cara al nostre treball s'ha de decidir bé el nivell de modularitat i dependències dels mòduls.

COM ESTRUCUTUREM LLAVORS?

Començant pel nivell més baix crearem una capa d'accés a base de dades i persistència de la manera més independent possible vers el tipus de bbdd que sigui. El motiu és que per exemple un client pot tenir una mysql que vol aprofitar i un altre una Oracle.

El projecte ha de contruirs-se utilitzant una eina de suport per al desenvolupament de web app amb patró MVC que sigui el més independent possible del contenidor d'aplicacions i el servidor d'app. Serà l'estructura general de qualsevol instancia del projecte i dependrà de la llibreria del framework.

Hi ha una primera pantalla de login que serà comú a qualsevol client que ho implementi. Conté un arxiu de vista i un de control. Tindrà referència al paquet bbdd.

La composició de l'aplicació és un menú mitjançant pestanyes on cadascuna d'aquestes té un dels mòduls de l'api. Aquesta part gestiona el canvi de tabs, si s'han de comunicar, si s'han de guardar canvis al canviar de tabs. Conté un arxiu de vista i un de control .Té dependència amb el paquet bbdd ja que mira a la mateixa sobre quines pestanyes te privilegis el usuari i per tant si pot accedir o no.

Una capa de seguretat amb una sèrie de filtres per controlar diverses qüestions com: intents d'intrusió via URL, timeouts, control d'URLs que accedeixen (identificació i/o limitació d'un numero concret). Es un paquet independent excepte si es volen guardar estadístiques o informes a bbdd.

Per dissenyar els mòduls els dividim en paquets (carpetes dins del projecte). Ara bé, podem tenir un arxiu de interfase d'usuari o un per cada component del mòdul. De la mateixa manera, podem tenir un arxiu de control per cada component o tots en el mateix. No obstant, els models de dades (ex. Camps recollits d'un formulari) han de ser individuals per cada component. En cas de tenir un arxiu de vista i un de control per mòdul, aquests tenen dependència amb el paquet de bbdd i si referències a altres mòduls en cas de que s'hagin de comunicar (ex. Un formulari a un mòdul modifica una vista en forma de taula d'un altre). A l'apartat de implementació es proven les dues maneres i es determina quina és la mes adient per aquest projecte.

4.4 Arquitectura del sistema

- L'arquitectura lògica defineix de quina manera està distribuït el software per packages, capes i subsistemes. D'esquerra a dreta observem separades per línees negres la capa de presentació, de negoci i de model respectivament a la *figura 13*.

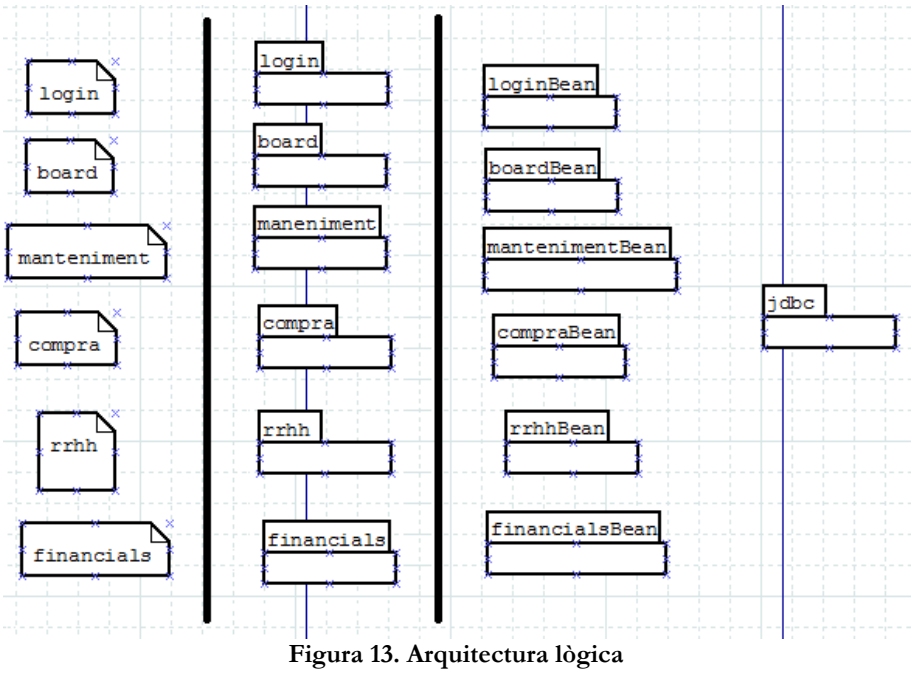


Figura 13. Arquitectura lògica

- L'arquitectura física determina que parts tindrem al client (part esquerra) i quines al servidor (part dreta) *figura 14*.

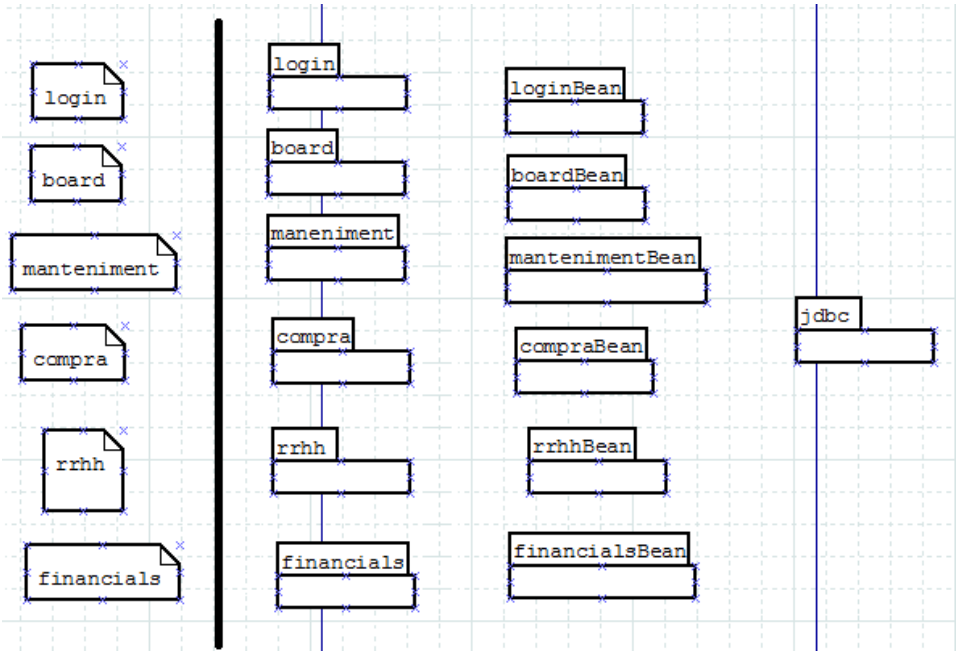


Figura 14. Arquitectura física

4.5 Extensió d'UML per modelar aplicacions web

Al plantejar-nos el dissenyar el model d'una aplicació web observem que el UML estàndard no s'ajusta del tot al tipus de elements, classes i relacions entre elles d'aquest tipus de apps.

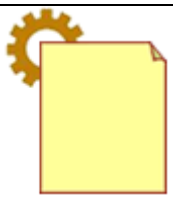
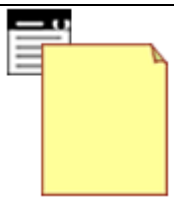
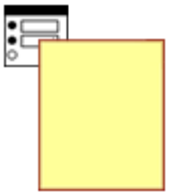
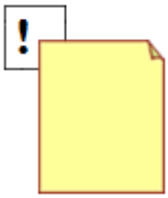
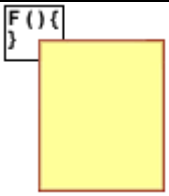
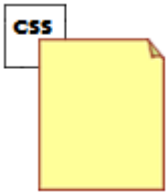
Després de cercar a la web, descobrim que, efectivament, els estereotips i relacions típics no són usats en aquest àmbit. Com a solució trobem una extensió de UML anomenada WAE (Web Application Extension).

En 1998, Jim Conallen (enginyer de software del departament de models relacionals d'IBM) va definir aquesta extensió per tenir les eines necessàries per a la modelació d'aquestes apps.

Dels nombrosos diagrames que tenim a l'abast amb UML, WAE, es centra en definir el diagrama de classes i el diagrama de seqüència. Pels altres s'utilitzen els habituals.

Utilitzem Star UML amb el mòdul extensió Profile d'un projecte del departament de sistemes d'anàlisi de la universitat Simon Bolívar University of Venezuela.

Degut a que utilitzarem JSF com a MVC, les definicions de l'extensió no filen del tot prim, per això definim la següent llegenda:

Estereotips per a les classes	
Estereotip	Descripció
 Server Page	Pàgina del servidor encarregada de les tasques de negoci (control dels components, càlculs i operacions i connexió amb sistemes externs com ara la bbdd).
 Client Page	Pàgina de client o de presentació, habitualment codificada amb html i interpretades als navegadors del client. Es comunica amb les pàgines del servidor.
 **Form**	Col·lecció de camps d'entrada/sortida de dades. Mapejada directament amb una pàgina HTML.
 **Alerts**	Finestres emergents d'error, quadres de text indicatius o indicadors d'informació.
 **ClientScriptObject**	Scripts en JavaScript o jQuery que s'executen a la part client i donen dinamisme a la plana.
 **StyleScript**	Scripts amb css que gestionen els estils i la maquetació dels components que estan a les planes xhtml.
Estereotip per a les relacions entre classes	
Request	Una pàgina client sol·licita a una servidor certa lògica per determinar quin fluxe seguir. (ex. un usuari clica a un link i s'ha de

	comprovar si té permisos).
Submit	Acció de recollir i processar l' informació d'un form per part d'una pàgina de servidor.
Redirect	Si un mètode de la capa lògica determina que s'ha de navegar a una altra pàgina es produeix un redirect.
Render	Afegir component a una vista.

4.6 Login, navegació i seguretat

La pàgina login té associat el formulari així com el missatge de credencials obligatòries, estils, control de focus i reajustament de la grandària de la pantalla en cas de que el navegador es minimitzi evitant scroll per donar interfase d'aplicatiu.

La pàgina login del servidor té la funció de verificar les credencials contra la bbdd i generar un missatge d'error en cas de no ser correctes.

La pàgina main del client conté totes les pestanyes (Board, Manteniment,..) del menú i fa un request al servidor per consultar quines tindrà visible l'usuari.

La pàgina main del servidor accedeix a la bbdd i mira sobre quines pestanyes té privilegis l'usuari (*figura 15)*.

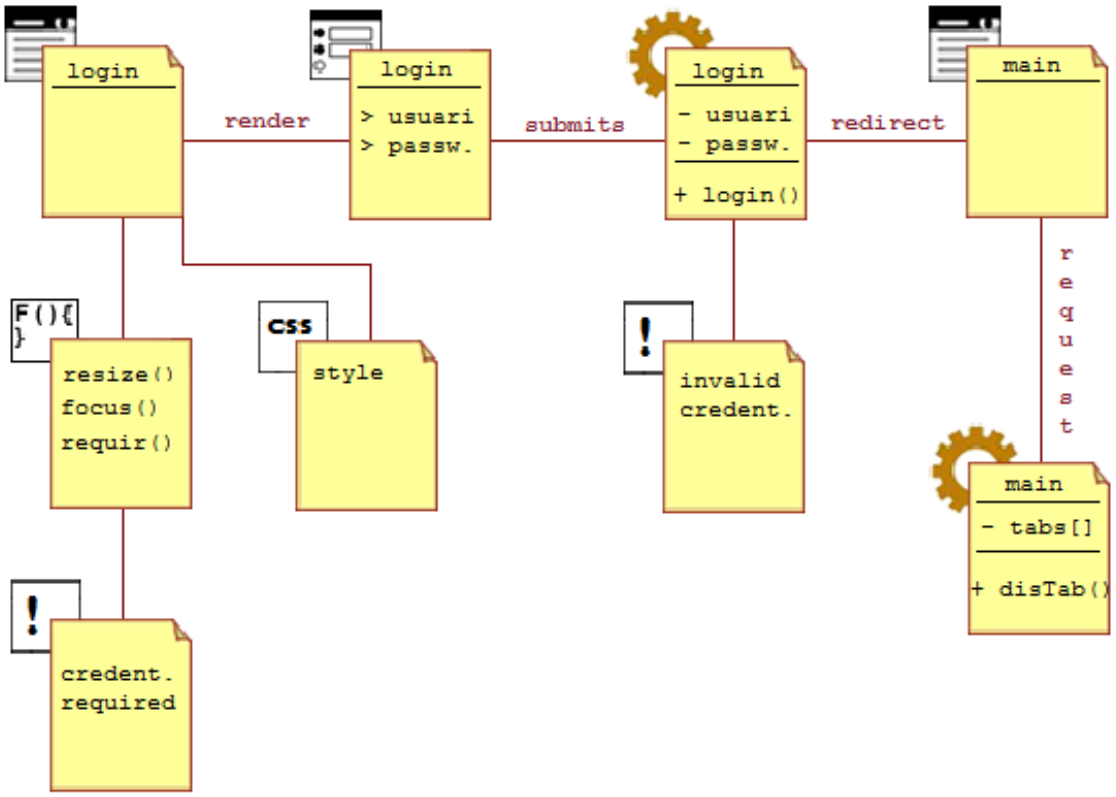


Figura 15. Diagrama de classes del Login

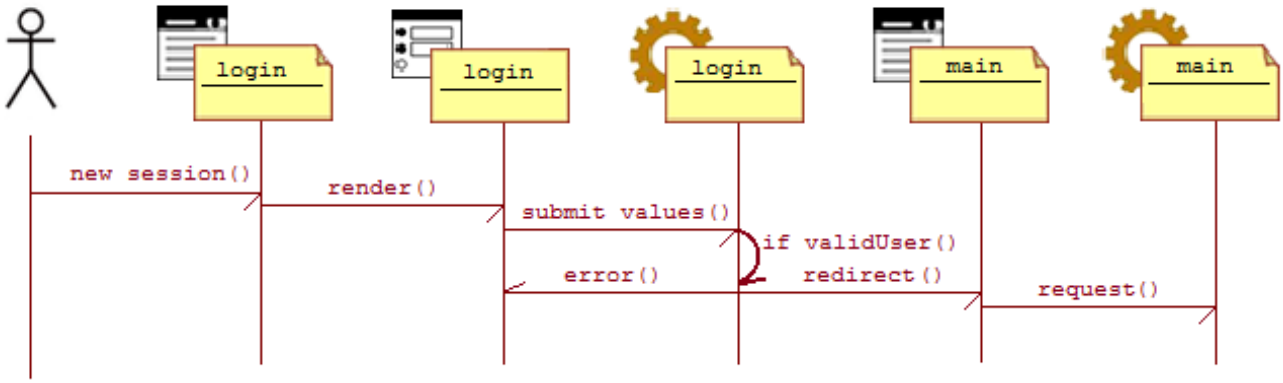


Figura 16. Diagrama de seqüència de Login

Ara presentem l'estructura de la bbdd (base de dades):

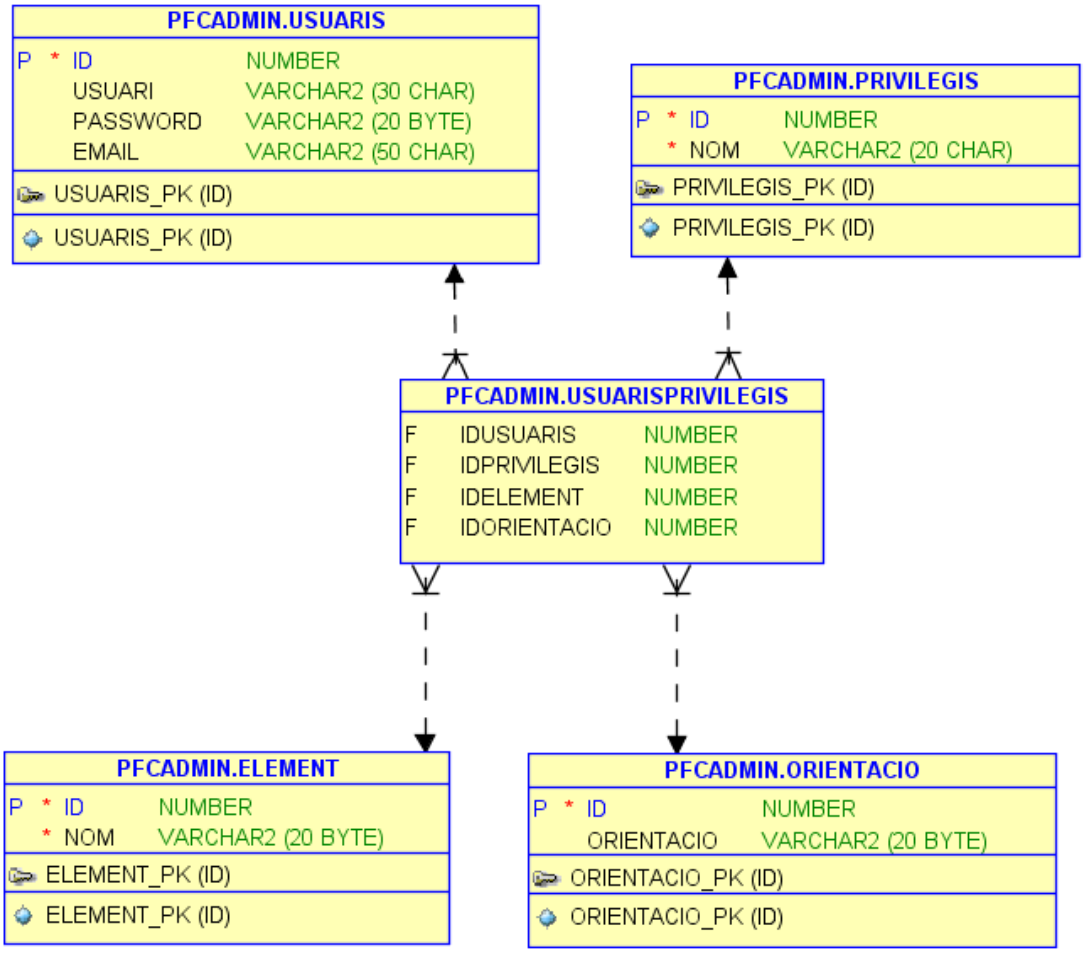


Figura 17. Bbdd de Login

Aquesta estructura ens permet que un usuari tingui privilegis a nivell de component ja que guardem l'element, l'orientació i els privilegis vers el component.

4.7 Board

La pàgina de Board compta amb el chart (indicador) que ens indica la capacitat de la bbdd i els tablespaces usats. Mitjançant un submit a la pàgina que té associada al servidor extreu aquestes dades de la bbdd *(figura.18,19)*.

D'altra banda schedule és un calendari de tasques que compta amb events (afegir, modificar i moure). Els events són gestionats a la part servidor. També al clicar a sobre renderitza un formulari d'entrada/edició de dades *(figura.18,19)*.

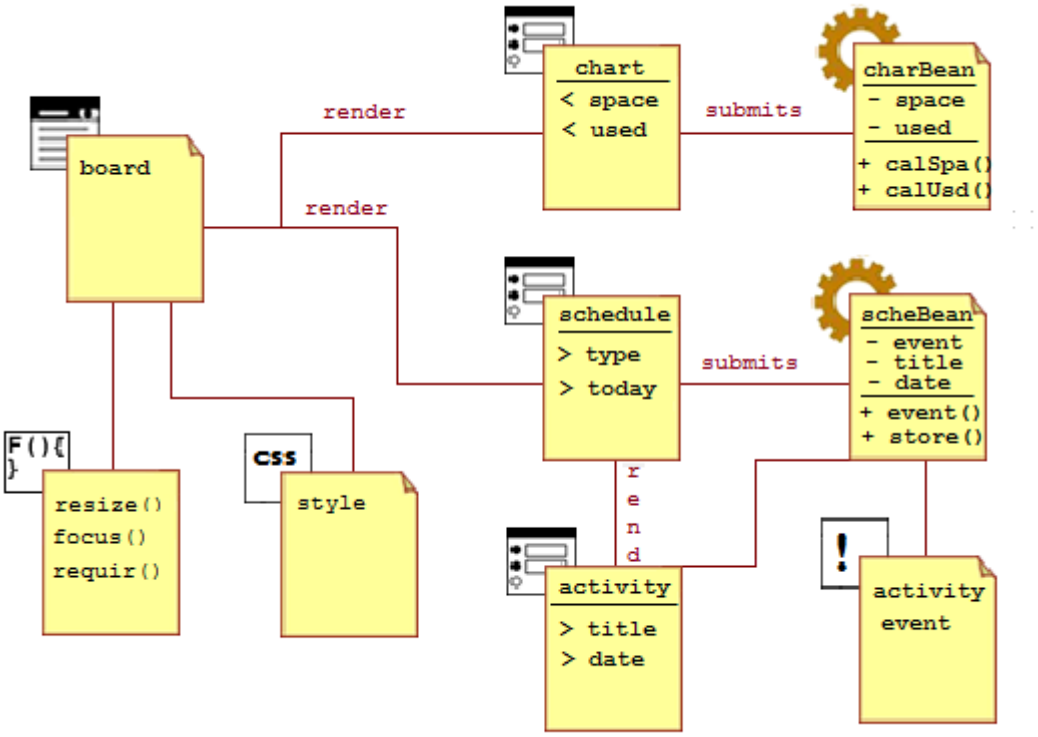


Figura 18. Diagrama de classes de Board

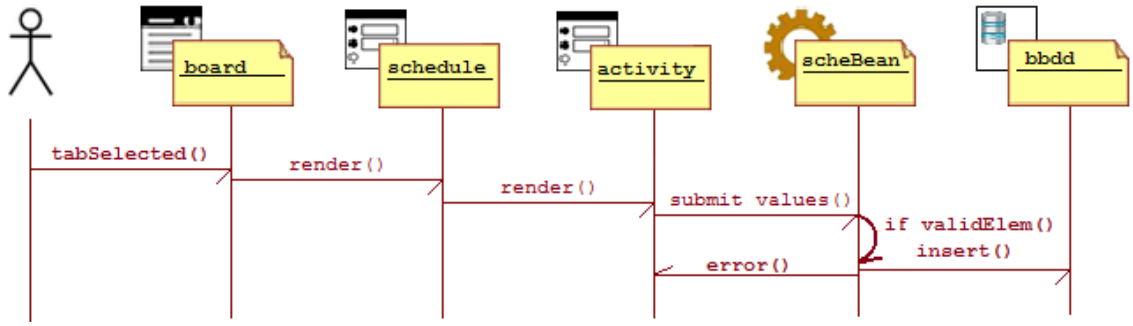


Figura 19. Diagrama de essència de Board

4.8 Manteniment

La pàgina de manteniment renderitza un acordió amb un resum de cada zona amb els seus punts de manteniment, punts de llum, linees i elements. Tot això es extret de la bbdd per la part servidor *figura 20*.

Per altra banda tenim un mapa on podem seleccionar (utilitzant desplegables) les zones, punts de manteniment, punts de llum individualment o en conjunt *figura 20*.

Del map renderitza una finestra per afegir components que associem a un boto *figura 20*.

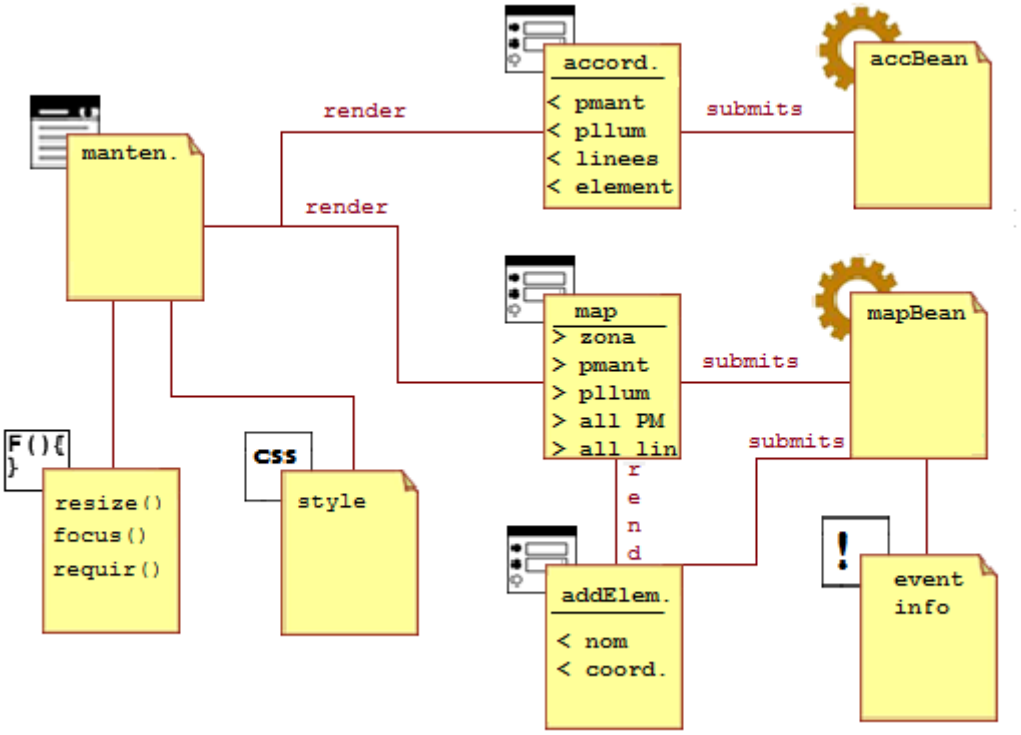


Figura 20. Diagrama de classes de Manteniment

Per concretar descrivim els cassos d'ús:

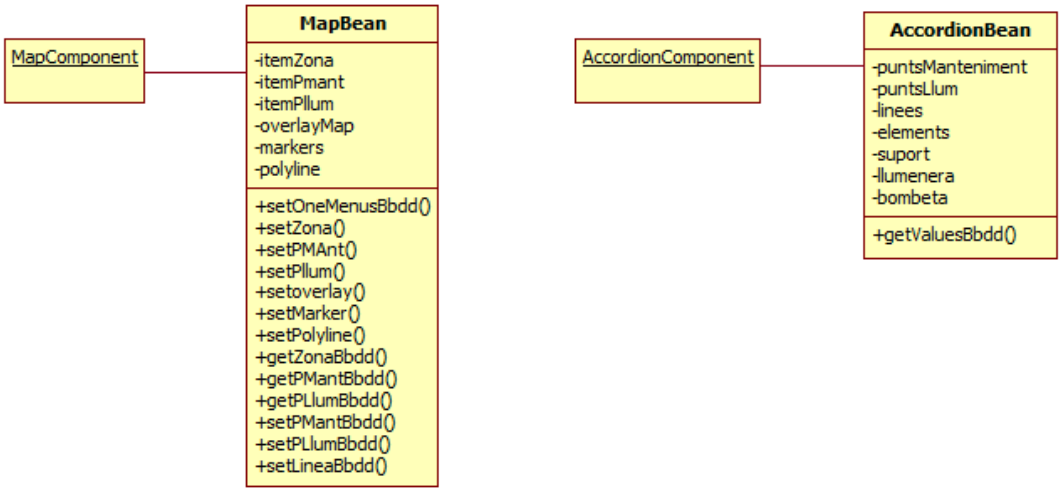


Figura 21. Detall lògica de manteniment

Cas d'us 1: Buscar PuntLlum al mapa

Actors: Usuaris: EmpleatCompres, EmpleatManteniment, EmpleatRecursosHumans

Propòsit: Permetre a un usuari amb privilegis sobre aquesta pestanya el buscar punts de llum i els elements relacionats amb el mateix (linees, puntManteniment, zona)

Preeconcions: Usuari validat i amb permisos, que existeixin punts de llum a la zona cercada

Sub-fluxes: Veure tots els punts de manteniment (boto PMant) i veure totes les linees (boto Linees) després d'haver escollit zona.

Excepcions: No s'ha escollit zona

La zona no té punts de manteniment

No s'ha escollit pManteniment

No hi ha punts de llum associats al pManteniment

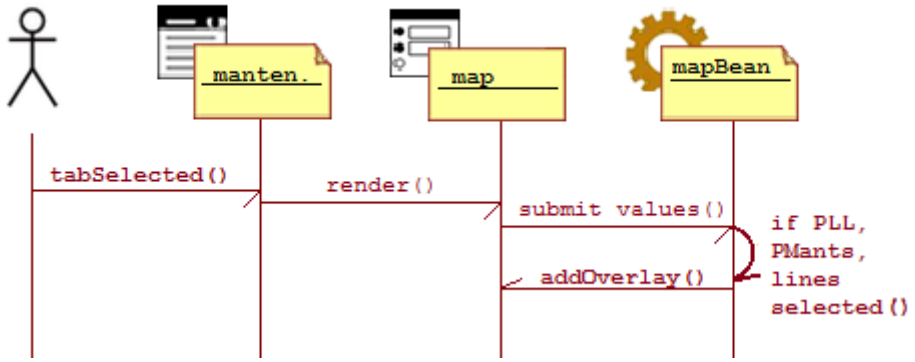


Figura 22. Diagrama seqüència de manteniment cas1

Cas d'us 2: Insertar element al mapa

Actors: Usuaris: EmpleatCompres, EmpleatManteniment, EmpleatRecursosHumans

Propòsit: Permetre a un usuari amb privilegis sobre aquesta pestanya afegir punts de llum o punts de manteniment

Preconcions: Usuari validat i amb permisos

Sub-fluxes: Afegir punt de manteniment o línea

Excepcions: No s'ha introduït una coordenada correcta

Falta un camp obligatori

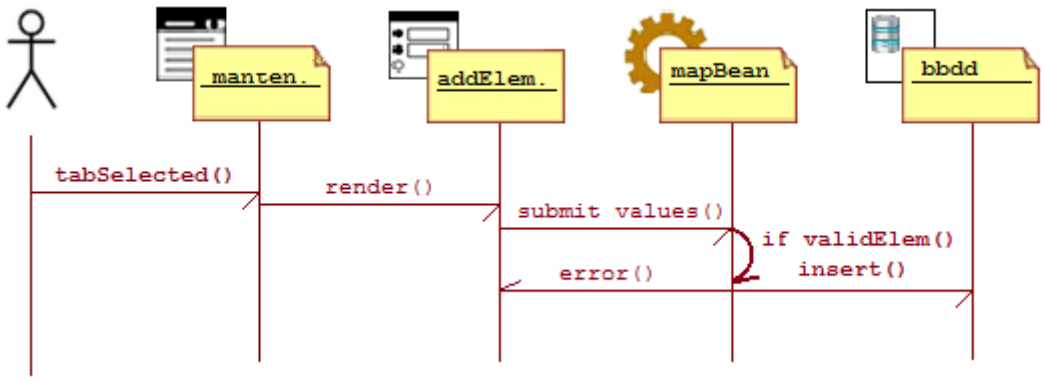


Figura 23. Diagrama seqüència de manteniment cas2

A la pàgina següent trobem les parts de la bbbdd implicades a aquest mòdul. Les llistem i expliquem a continuació:

- *Figura 24*. Fa referència a l'organització territorial descrita a l'apartat 2.2. Delegacions amb un o varis contractes que a la seva vegada tenen una o varies zones.
- *Figura 25*. Referent als punts de llum associats a un punt de manteniment distribuït en una línea amb d'altres.
- *Figura 26*. Referent als elements que composen un punt de llum (suport, llumenera i bombeta).

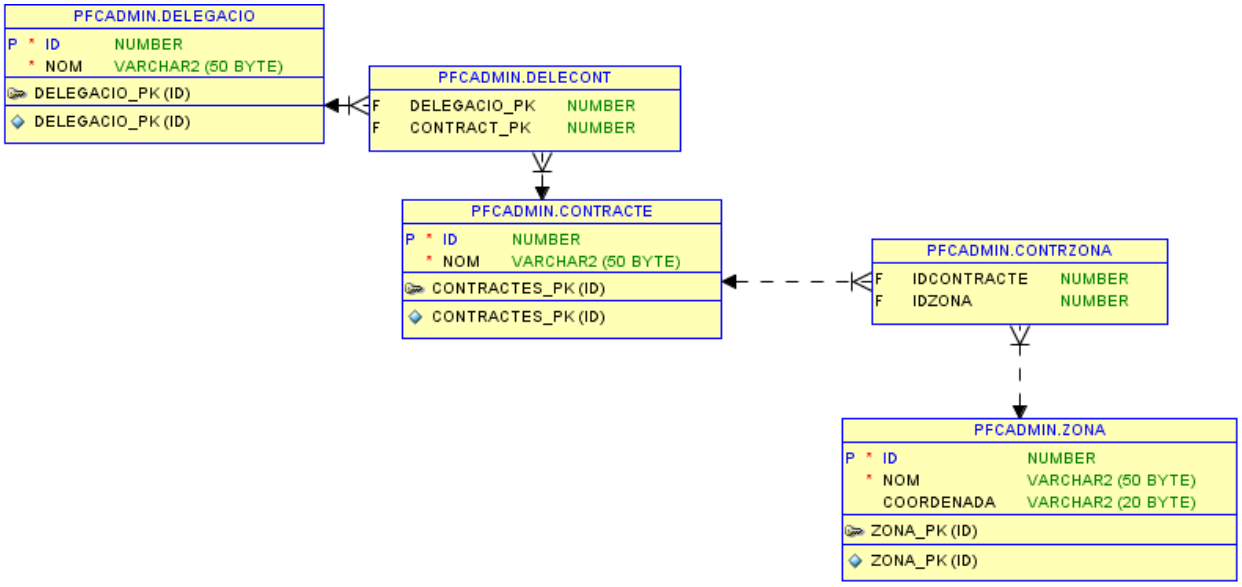


Figura 24. Bbdd de Manteniment (zones)

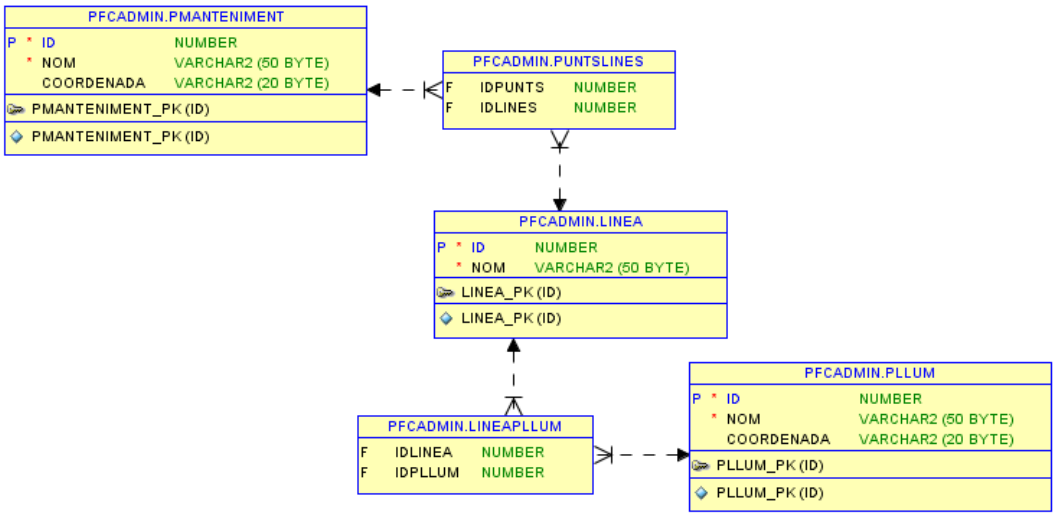


Figura 25. Bbdd de Manteniment (punts de manteniment)

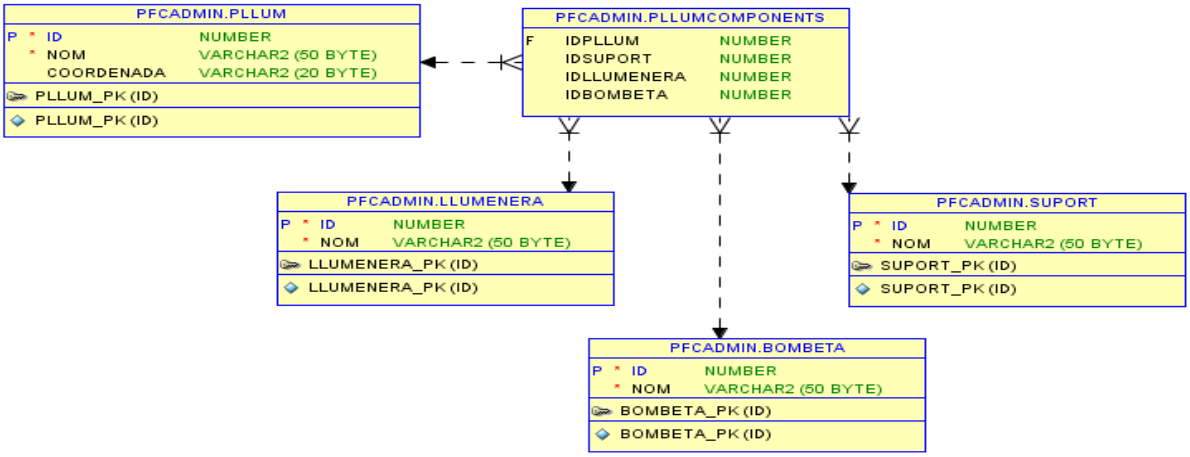


Figura 26. Bbdd de Manteniment (punts de llum)

4.9 RRHH

La pàgina de RRHH té com a element principal el component tableRow. En aquest s'extreuen les tasques de la pestanya Board utilitzant el mateix model de dades. *figura 27*. Les tasques es poden afegir, eliminar o editar. En aquest mateix apartat exemplifiquem un cas d'us.

Posteriorment es pot clicar sobre export (botó) generant un arxiu de sortida que inclou la taula mencionada al paràgraf anterior *figura 27*.

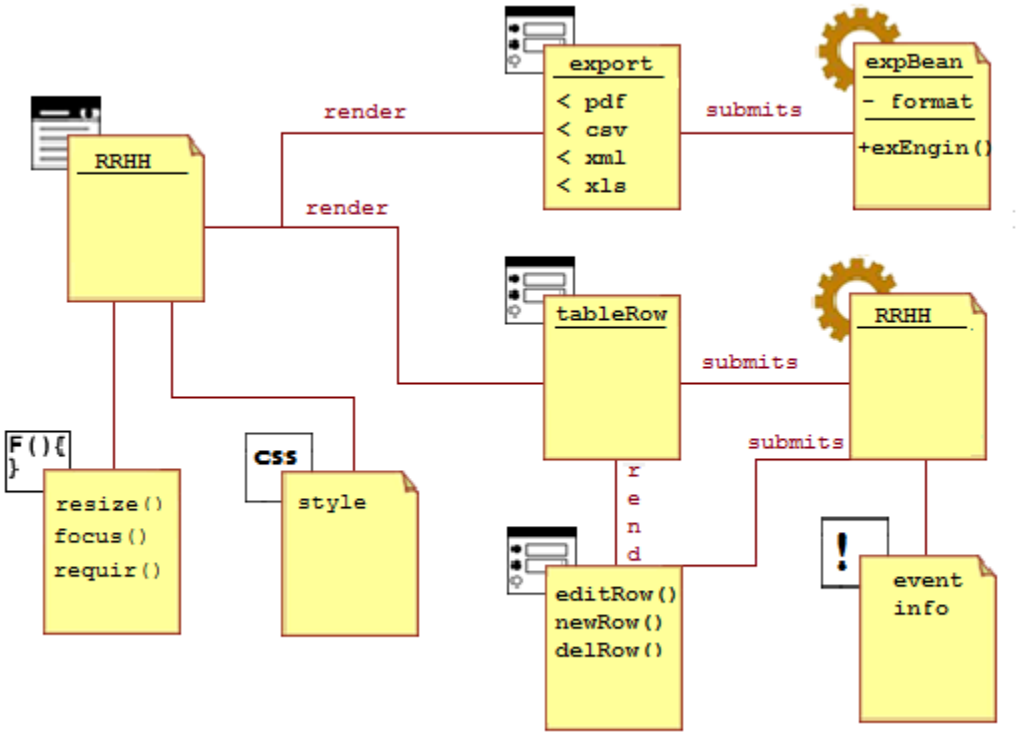


Figura 27. Diagrama de classe de RRHH

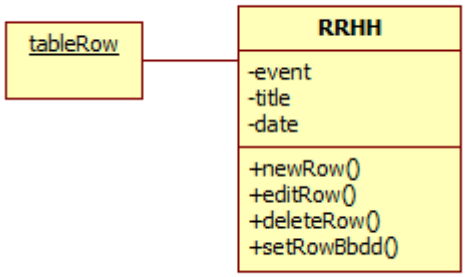


Figura 28. Detall lògica de RRHH

Cas d'us 1: Introduir nova fila

Actors: Usuaris: EmpleatRecursosHumans

Propòsit: Permetre a un usuari amb privilegis sobre aquesta pestanya introduir noves activitats a la taula

Precondicions: Usuari validat i amb permisos per a aquesta pestanya, que existeixin activitats programades

Sub-fluxes: Editar fila existent, eliminar fila

Les dades introduïdes no tenen el format adequat

Falta algun camp obligatori per introduir

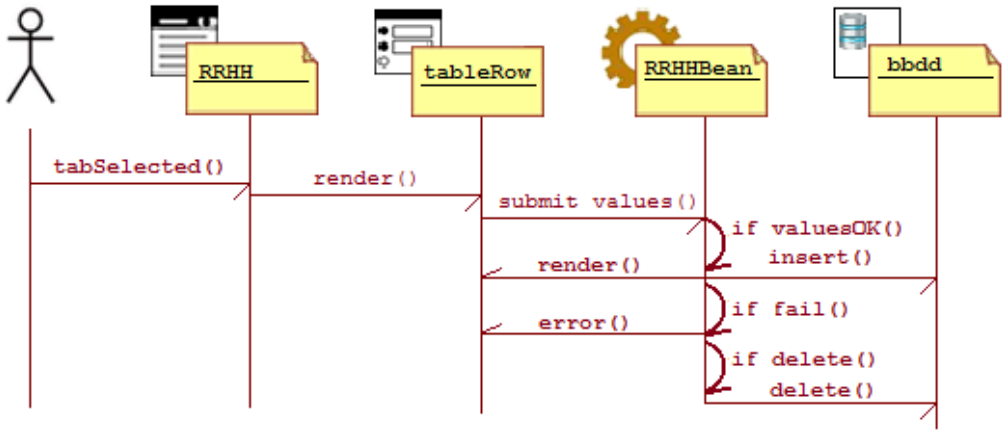


Figura 29. Diagrama seqüència cas 1

Cas d'us 2: Exportar les dades de la taula a un arxiu de sortida

Actors: Usuaris: EmpleatRecursosHumans

Propòsit: Permetre a un usuari amb privilegis sobre aquesta pestanya exportar les dades que està visualitzant a la taula en un arxiu

Precondicions: Usuari validat i amb permisos, que existeixin activitats a la taula

Sub-fluxes: Exportar en pdf, csv, xls o xml.

Excepcions: No hi han dades a la taula.

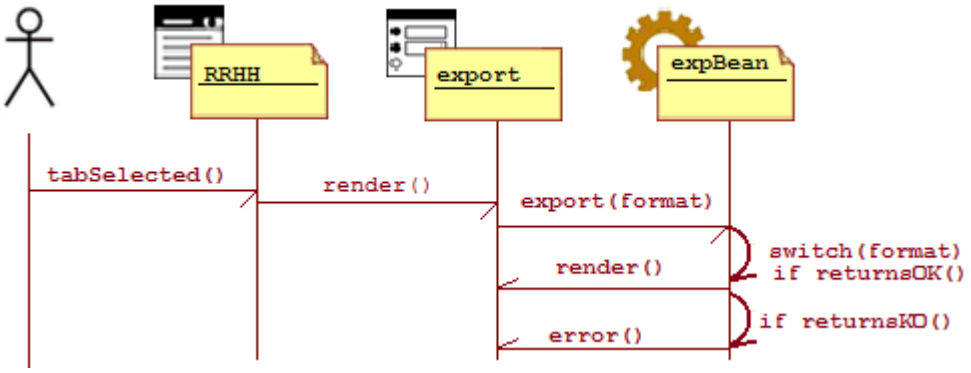


Figura 30. Diagrama seqüència cas 1

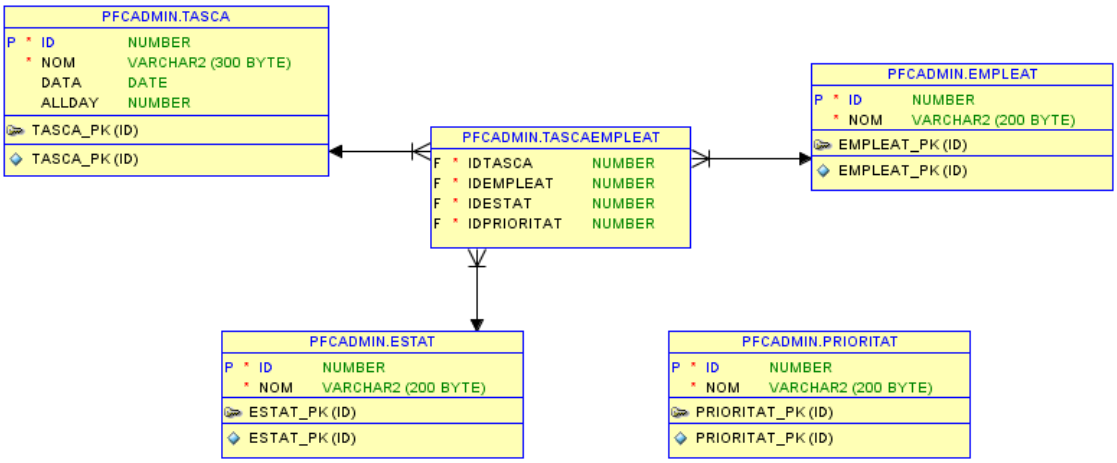


Figura 31. Bbdd de RRHH

5. Implementació

Si A la part de disseny s'estableix la direcció i els principis que es seguiran durant la gestació del software. A la part d'implementació es dona resposta a aquests principis amb la tecnologia escollida.

5.1 Contenidor JBoss

La primera tasca que s'ha de fer en aquest tipus de solució és l'instal·lació d'un servidor d'aplicacions. Això no és més que un sistema d'entrada i sortida de dades situat en una màquina diferent a la client que conté i executa aplicacions que s'estableixen en el mateix. El servidor es manté constant a les peticions que rep i respon amb format web. En el nostre projecte hem utilitzat una versió localhost, és a dir, s'ha d'imaginar que l'assignació virtual de recursos per al servidor web en l' implementació real suposaria una computadora diferent.

Els motius per utilitzar aquesta arquitectura y no per exemple una aplicació tradicional ubicada a la màquina client són:

- La disponibilitat: molts usuars han d'accedir a l'aplicació. Aquesta és una arquitectura que ens anirà bé per tenir disponible sempre el software i poder gestionar la concurrència entre els usuaris.
- Escalabilitat: el nostre servidor tindrà una capacitat límit de servir peticions. Amb aquesta metodologia podrem ampliar els recursos afegint nous blocs al servidor.
- Manteniment: al depurar, fer manteniment o dotar de noves actualitzacions al software, nomes caldrà fer-ho al servidor i automàticament els usuaris ho veuran al accedir al seu client sense actualitzar res.

En el nostre cas hem utilitzat Joss Rutime 7.1 i per facilitar l'operació, dotant a l'eclipse del plugin Jboss tools utilitzant el link *Add new server* només s'ha d'indicar el JRE i el path on està la descàrrega del mateix contenidor *figura 32*.

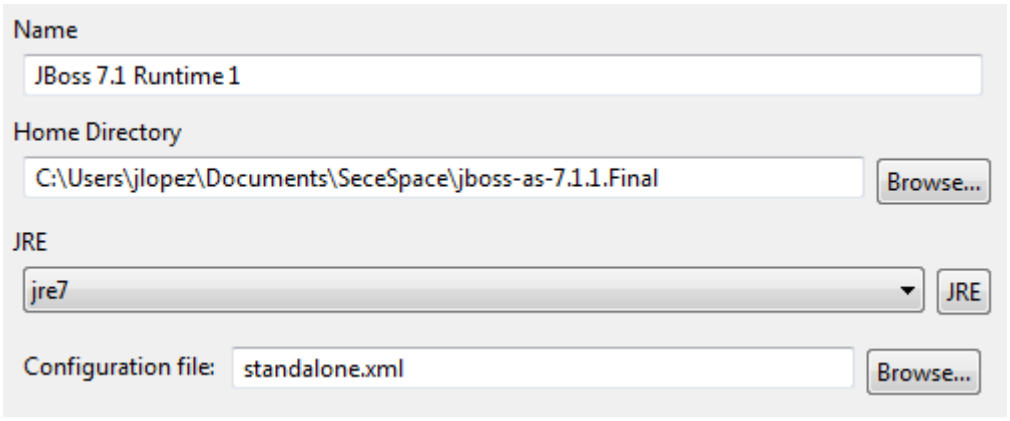


Figura 32. Instància servidor JBoss

5.2 Connexió amb base de dades JDBC i Oracle

Instal·lem una instància de Oracle Database 11g Enterprise Edition 32bits. Aquest software ens proporciona mitjançant una URL localhost un manager per administrar la base de dades. Creem un usuari mitjançant un assistent tenint en compte:

1. Assignar una password.
2. Assignar el tablespace USERS i el tablespace(NOTA PIE:AREA VIRTUAL ON S'EMMAGATZEMEN ELS OBJECTES) temporal TEMP.
3. Assignar rols:
   - CONNECT: permet connexió a BD y consulta de los dades dels esquemes del usuari. (CREATE TABLE, CREATE VIEW...)
   - RESOURCE: permet la gestió d'objectes del meu esquema (TRIGGERS, PROCEDURE...).
   - DBA: permet l'administració de la BD dotant amb privilegis de sistema (UNLIMITED TABLESPACE)
   - EXP_FULL_DATABASE concedeix SELECT ANY TABLE, BACKUP ANY TABLE, INSERT, DELETE, UPDATE
   - IMP_FULL_DATABASE privilegi BECOME USER
   - DELETE_CATALOG_ROLE, EXECUTE, SELECT per borrar, executar i consultar totes les taules y vistes del diccionari de dades (NOTA AL PIEconjunt de taules del sistema, de lectura, que

proporcionen informació sobre la base de dades (ex. Què taules té l'usuari i quines restriccions afecten a les columnes de les mateixes.)

Utilitzem Oracle SQL Developer per gestionar la connexió *figura 33*:

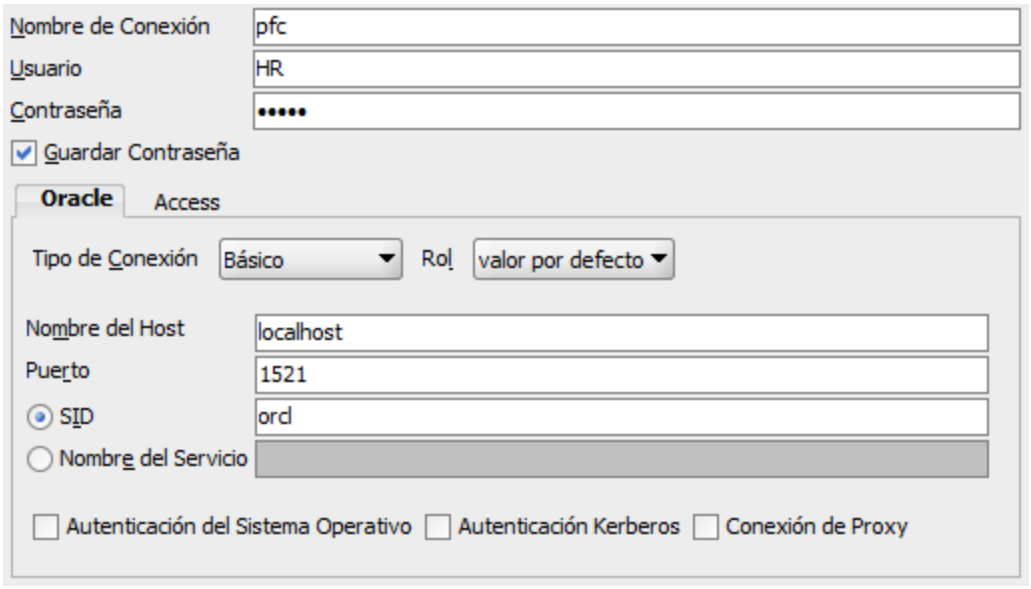


Figura 33. Arquitectura MVC

I connectem amb el següent codi jdbc:

```
package jdbc;

import  java.sql.*;
import  java.util.*;

public class JDBC {
  try
  {
    // Comprovar que existeix el driver
    Class.forName("oracle.jdbc.driver.OracleDriver");
  } catch (ClassNotFoundException e) {
    // Create a XMLOutputFactory
    XMLOutputFactory outputFactory = XMLOutputFactory.newInstance();
    XMLEventWriter eventWriter = outputFactory
```

5.3 Framework MVC JSF i tecnologia JSP

JSF (Java Server Faces) és un framework de desenvolupament basat en l'arquitectura MVC (Model Vista Control). La seva intenció és estandarditzar el desenvolupament d'aplicacions web (similar a l'estil de l'API Swing), on l'interfície es realitza mitjançant components i basada en events. No obstant el cicle dista una mica de frameworks com Struts i és important entendre com treballa.

Alguns conceptes que hem de tenir en ment són:

- Vista: conjunt de fitxers JSP i la descripció de components JSF que formen les pantalles d'interfase amb l'usuari de l'aplicació. Vinculen els components amb els Managed Beans (explicats a continuació).
- Model: fitxers managed Beans. Són POJO (classe simple que no depèn d'un framework) que s'associa amb els components de l' interfície d'usuari (UIComponent). Poden tenir validesa durant l'aplicació, una sessió de client, mentre dura una mateixa pàgina de vista o durant una el temps entre una petició i la resposta a un client. Es determina afegint una anotació a la classe Java (@ApplicationScoped, @SessionScoped, @ViewScoped, @RequestScoped respectivament). Dins de les seves funcions està:
    - Emmagatzemar referències als components de la vista i posar-los a la disposició del controlador a la part de client .
    - Proveir de les propietats necessàries per a la vista, com el comportament de la pantalla, la informació que es presenta i inclusivament fins a alguns elements del disseny gràfic.
    - Implementar els mètodes que poden ser usats pels components per comunicar el servidor amb l' interacció amb l'usuari (events) i la navegació.
- Controlador: Faces Servlet (definit a faces-config.xml), examina les peticions http i invoca els controladors d'events i navegació a través dels mètodes dels Managed Beans.

```
<h:commandButton label="Aceptar" action="#{loginControlador.verificarUsuario}"/>
```

```
String verificarUsuario() {
    if (...)
        return "exito";
    else
        return "fracaso";
}
```

```
<navigation-case>
        <from-outcome>exito</from-outcome>
        <to-view-id>/exito.jsp</to-view-id>
</navigation-case>
<navigation-case>
        <from-outcome>fracaso</from-outcome>
        <to-view-id>/fracaso.jsp</to-view-id>
</navigation-case>
```

Altres elements imprescindibles (aquests són interns i en desenvolupaments on no s'ha de generar components nous no s'han de manipular):

- Renderizador: generen la representació dels components JSF i recuperen les dades introduïdes pels usuaris.
- Convertidors: transformen els valors dels objectes Java(Integer, Double, Date,...) a valors dels objectes JSF(html) i viceversa.
- Validadors: comproven restriccions d'entrada (ex. Format vàlid)

El cicle de vida d'un projecte JSF és:

1. Localitzar (mirant la URL) la vista corresponent i generar-la.
2. En cas de tenir la vista emmagatzemada, mostrar si han hagut canvis.
3. Actualitzar el estat dels valors dels component a la part client.
4. Processar els convertidors i després els validadors.
5. Actualitzar els valors als managed Beans.
6. Aplicar la lògica sol·licitada a la part client.
7. Renderitzar la resposta.

Per generar un projecte JSF amb l'Eclipse fem:

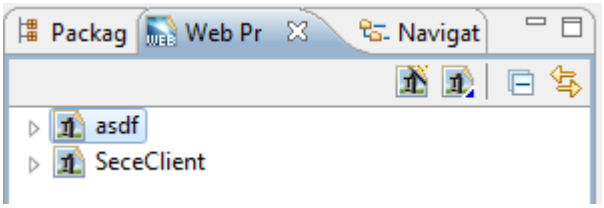


Figura 34. Arquitectura MVC

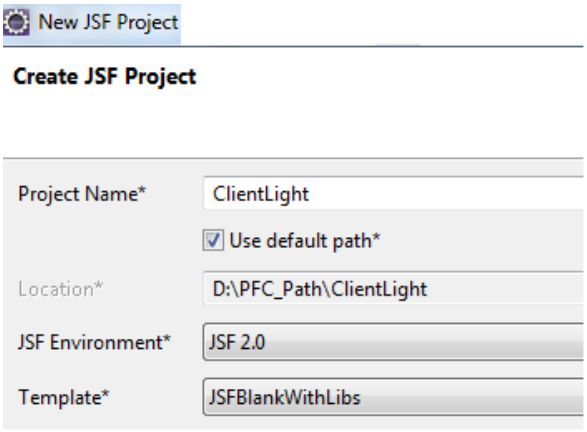


Figura 35. Arquitectura MVC

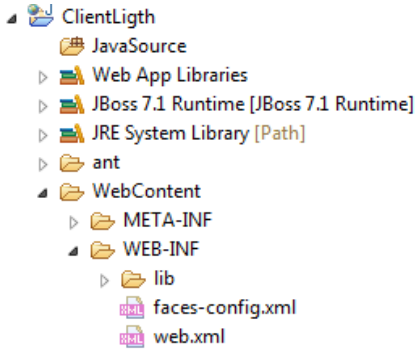


Figura 36. Arquitectura MVC

5.4 Components Primefaces

Al utilitzar els components de JSF, ens adonem que la part visual queda bastant simple respecte al tipus de webs que veiem avui en dia. Per això s'ha dut a terme una investigació per millorar-ho. A Internet trobem diverses llibreries ,que a partir dels components bàsics de JSF, els enriqueixen mitjançant JavaScript. El desenvolupador ha d'afegir aquesta llibreria i cridar els nous components mitjançant tags. És a dir si amb JSF pla utilitzàvem <h:comandbutton> amb la nova nostra llibreria fem <p:commandbutton>. Així utilitzem un component que deriva d'un dels originals però que incorpora nous efectes visuals, funcionalitats, ...

De les diverses opcions del mercat les finalistes van ser: RichFaces i Primefaces. Com no coneixíem cap vam fer unes probes amb cadascun, mirar les opinions dels programadors a la web i quina comunitat de suport era mes activa. Finalment ens vam decidir per Primefaces degut a que ens va ser mes fàcil de posar en marxa (només copiant la llibreria ja està llest per usar-ho), les opinions tant de programadors com empreses era molt abundant i positiva i té un espai de suport gratuït molt ràpid i eficaç.

Tot i que a continuació descrivim un exemple de com es treballa amb aquesta llibreria, a la web primefaces que es descriu a la webgrafia, es pot observar un exemple d'implementació d'un component complet mitjançant un codi comentat.

Tenint en compte que la llibreria es troba al build path del projecte i ha sigut desplegada amb èxit, mitjançant un arxiu html declarem la vista (utilitzant els tags que fan referència als components):

```
<p:growl id="growl" showDetail="true"/>
<p:calendar value="#{prova.date1}" mode="inline" id="inlineCal">
```

Com podem observar el primer tag té un atribut showDetail. Aquest component ens permet treure per pantalla una alerta. La decisió de quan ha de mostrar-se una alerta es fa a la part servidor. El tag calendar té com a valor un índex. Aquest índex apunta a un arxiu .java que és la part de negoci.

```
private Date date1;
public void setDate1(Date date1) {
    this.date1 = date1;
}
```

Un cop que fem deploy al nostre servidor d'aplicacions podem veure els resultats.

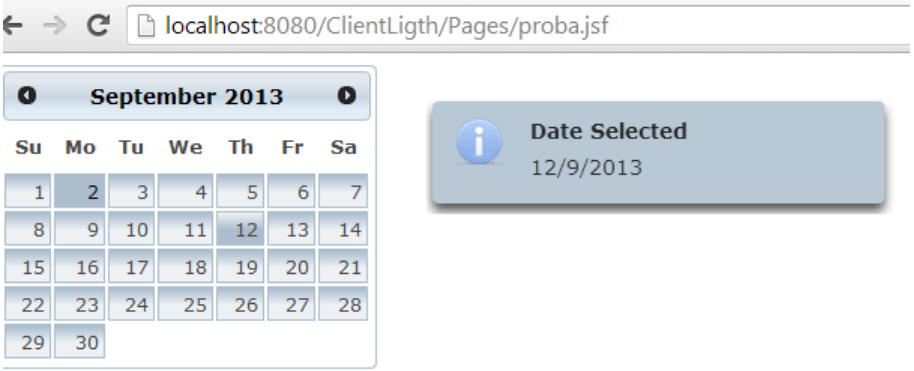


Figura 37. Arquitectura MVC

5.5 Seguretat; login i filtre URL

Al estar treballant sobre entorns web, un dels temes més importants és la seguretat. Per això es va plantejar un estudi dels atacs més comuns i si la nostra tecnologia podia respondre i evitar els mateixos. Finalment vam decidir que els següents eren les vulnerabilitats que més havíem de tractar:

- Injecció SQL → consisteix en introduir codi intrusiu al nivell de validació de entrades de credencials. Així per exemple a un formulari d'usuari l'atacant introduiria una consulta sql que generaria una query destructiva. Al traduir-se a la capa d'accés de base de dades quedaria una query que faria malbé la bbdd com es mostra a continuació:

```
1)  SELECT * FROM usuarios WHERE nombre = 'Alicia';
1b) SELECT * FROM usuarios WHERE nombre = '" + usuari + "';"

2) SELECT * FROM usuarios WHERE nombre = 'Alicia';
   DROP TABLE usuarios;
   SELECT * FROM datos WHERE nombre LIKE '%';

SOL) SELECT * FROM usuarios WHERE nombre = ?;
     Statement.setString(1, nombre Usuario);
```

A la sentencia 1) en blau podem observar el que un usuari no atacant introduiria i en verd la query preparada per ser executada (utilitzant el model 1b). A la sentencia 2) en vermell veiem que és el que introdueix un atacant i juntament amb el verd la query que es tradueix i que eliminaria la taula d'usuaris.

La solució que hem proposat (SOL) és la de parametritzar els SQL, és a dir que amb una funció (setString) introduirem l'entrada. Si a la solució l'introduïm la part vermella del punt 2) observem que la query no funciona degut a que queden unes cometes obertes dins la sentencia.

- Cross Site Scripting → igual que en el punt anterior, l'atacant intentarà enviar mitjançant formularis o altres components codi que al compilar comprometi el sistema. En el nostre cas seria posar per exemple a un quadro d'entrada un codi javaScript (<script>bucle infinit</script>). Per solucionar-ho s'ha de parsejar cada cop que s'interpreta el codi, cosa que s'escapa del scope del projecte. Per això hem utilitzat una llibreria que disposa de diferents estàndards per parsejar (argument policy) i només l'hem de cridar introduint pel argument input el codi a comprovar:

```
CleanResults cr = validator.scan(ESAPI.encoder().canonicalize(input),policy);
```

- Robatori de sessió → és mitjançant per exemple un dels mètodes anteriors, l'atacant situa un link redirigit a una pàgina seva. Posteriorment mira els logs del seu servidor aconseguint l'ID de sessió. Amb aquesta id accedeix mitjançant URL com si fos el client logat. Per això hem pres tres mesures.
  - Utilitzar URL's que no mostren valors de connexió (a part de que es visualitzen de manera més neta).
  - Quan un usuari no fa moviment, un timeOut() de 5 minuts tanca la sessió havent de fer l'usuari posteriorment login de nou.
  - Utilització d'un filtre que detecta si l'usuari ha passat pel login *figura 38*.
- Accés per URL → el encès a la URL redirigeix a la pagina de login mitjançant la definició de la welcome-file al fitxer web.xml. el filtre mencionat al punt anterior redirigeix al Login() qualsevol usuari no logat.

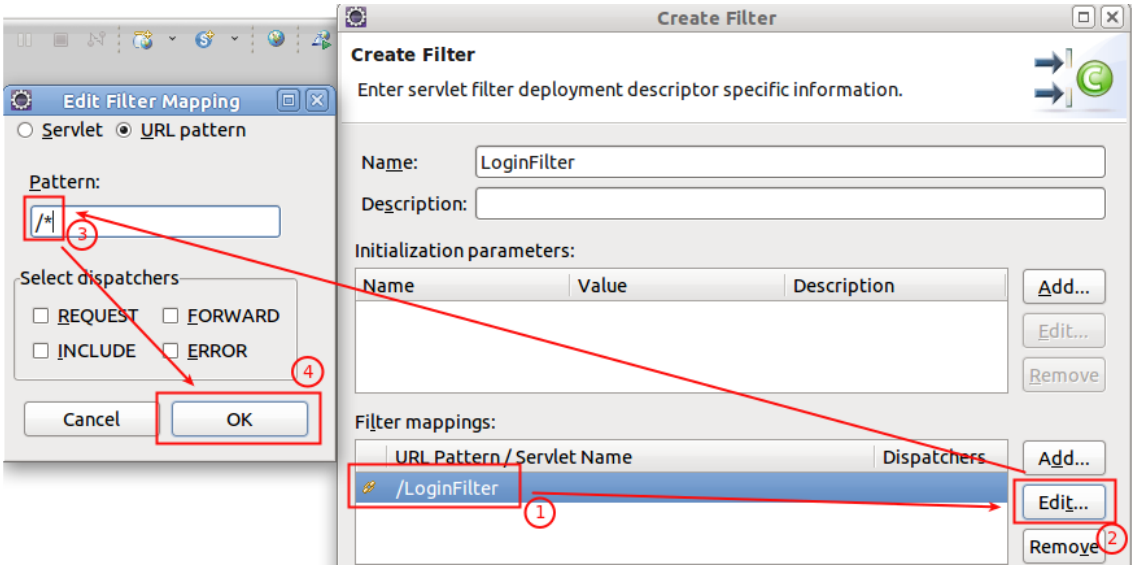


Figura 38. Arquitectura MVC

```
//Si l'usuari no està logat es redirigit al login
if (loginBean == null || !loginBean.estaLogeado()) {
  System.out.println("is not logged");
       res.sendRedirect(req.getContextPath() +
       "/Pages/login.jsf");
  return;
       }
```

5.6 Estils i contingut dinàmic CSS i JQuery

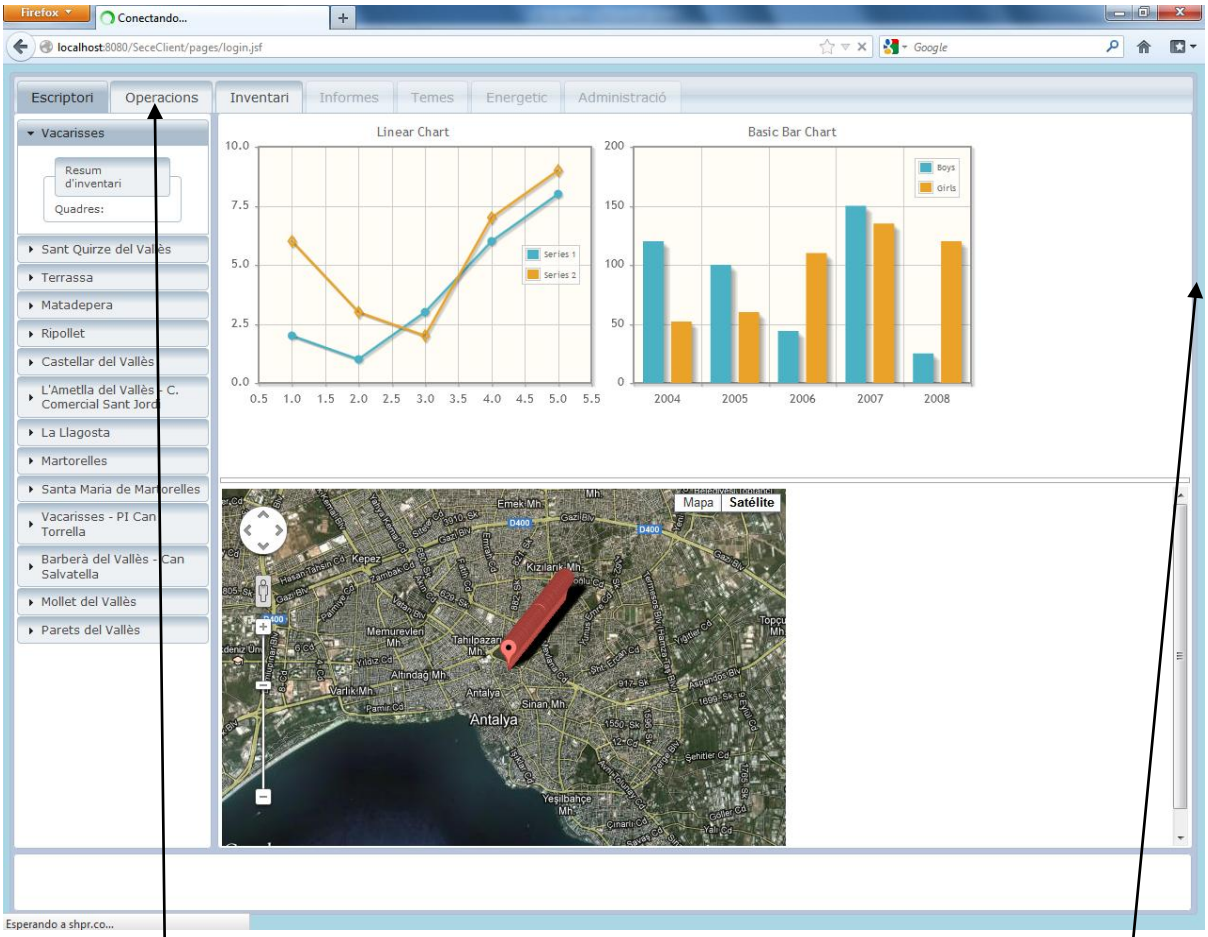
Al parlar d'una aplicació web de nova generació, no podem passar per alt la part de disseny i dinamisme de components. Si bé ja hem parlat de que la llibreria Primefaces utilitza jQuery i CSS hem comprovat que ofereix la possibilitat de personalitzar al client el seu producte.

Efectivament, degut a que els components de la llibreria de presentació estan inclosos dins de <h:form> i podem dividir-los amb podem utilitzar els ids d'aquests dos tags. Com les pàgines són html estàndard ens permet situar a les capçaleres <script> amb allò que vulguem que sigui interpretat. En el nostre cas hem provat:

- El fons (background) del login.
- El formulari de login sacseja si les credencials són incorrectes.
- Els charts són interactius.
- Al scheduler es poden moure les dates arrossegant amb el ratolí

- Al mapa s'actualitza amb jQuery els mapes i els outputs.
- La taula es pot editar manual i dinàmicament.
- Navegació TabMenu

5.7 Navegació TabMenu



Tenim un menú de pestanyes per navegar a través de l'aplicació.

L'app té aspecte d'aplicació d'escriptori ja que es desactiva el scroll característic a portal web.

Figura 39. Arquitectura MVC

5.8 Charts

Els charts són gràfics que ens mostren sortides de dades però de manera gràfica i comprensible. En el nostre cas hem instal·lat a la pestanya de Board un parell:

- Total d'espai utilitzat de la bbdd.
- Espai utilitzat a cada TableSpace.

Com hem destacat abans de cada component trèiem el seu identificador i li podem assignar propietats dinàmiques. Per demostrar-ho al ClientLight, als gràfics implementats veiem que està l'event on-mouse (s'activa un event que canvia de color la barra sobre la que ens posicionem i indica el valor exacte). Per altra banda al clicar a la llegenda podem escollir l'opció de veure un tipus de barra (ex. les barres blaves o taronja) o no veure cap *figura 40*.

Associats als charts hi ha unes tasques de servidor que cada dia (en un horari on no s'accedeix a la bbdd) s'executa una query que actualitza els components. A continuació el aspecte del component:

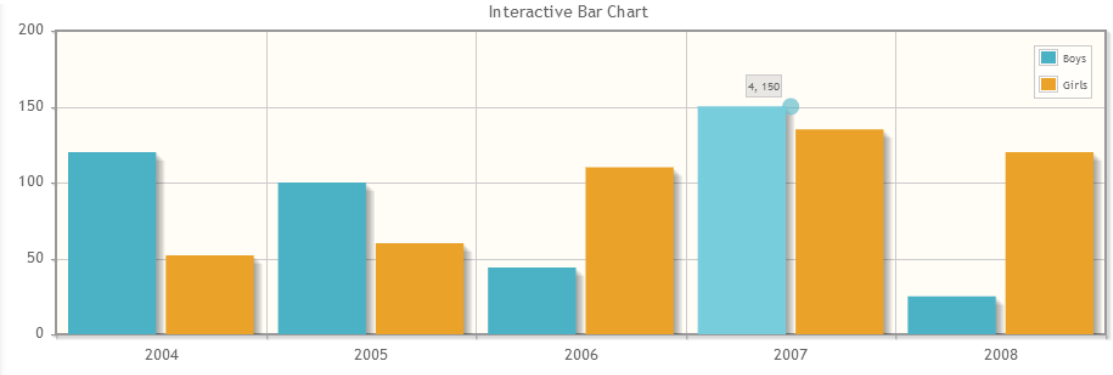


Figura 40. Arquitectura MVC

5.9 Schedule

Es schedule és una mena de taula amb forma de calendari. El propòsit del component es tenir control i poder organitzar les tasques a realitzar (sobre els actius).

A dit component l'hem incorporat alguns botons amb events:

- Vista mensual, diària i setmanal.

- Botó today: al clicar-ho situa el calendari en el dia actual.

Al clicar sobre un dia podem afegir una tasca. De la mateixa manera al clicar sobre una tasca, la podem editar *figura 41*.

Podem arrossegar les tasques i s'editen automàticament.

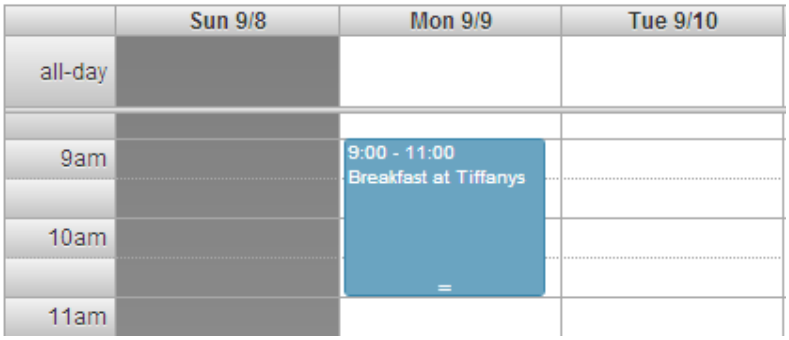


Figura 41. Arquitectura MVC

5.10 Maps

Aquest component és una extensió de la llibreria de JavaScript v.3 per a GMaps. El que ens proporciona és que només incloent un tag <p:map> al nostre codi html puguem visualitzar un mapa. Tota la gestió d'afegir overlays com markers, lines,... es fa des de la part java amb mètodes que ens posa al abast la mateixa llibreria primefaces *figura 42*.

Figura 42. Arquitectura MVC

No obstant, fent un anàlisi crític d'aquest component (que és en el que més temps s'ha hagut d'invertir), la llibreria té mancances respecte a les necessitats actuals d'aplicació web. Primerament, a l'hora d'afegir qualsevol element (marker, circle..) és senzill, es crea una instància (marker per exemple) des de la pàgina java de negoci i es crida el mètode addOverlay. Ara bé si hem de continuar afegint overlays (com és el nostre cas) comencen els conflictes. Per altra banda, la llibreria no incorpora removeOverlay(). Com a solució hem proposat cridar el mapa amb el tag <p:map> i mitjançant el seu identificador, a la mateixa plana html utilitzar directament la llibreria JavaScript.

5.11 DataTable

DataTable és un component tipus graella similar als que s'utilitzen a qualsevol interfase. Igual que els altres components, l'hem establert per a que pugui ser dinamitzat amb les eines que utilitzem al nostre entorn (events jQuery) *figura 43*.

La peculiaritat però es que apart de ser editable i amb possibilitat d'anular files, hem aconseguit que es comuniqui amb la pestanya Board. Essent així al introduir tasques a Board aquí les podem visualitzar i gestionar en el context de recursos humans.

La tècnica emprada ha estat el utilitzar la mateixa estructura de dades per ambdues pestanyes, així, i tenint en compte de que només podem visualitzar una a la vegada tenen dita estructura com a element compartit.

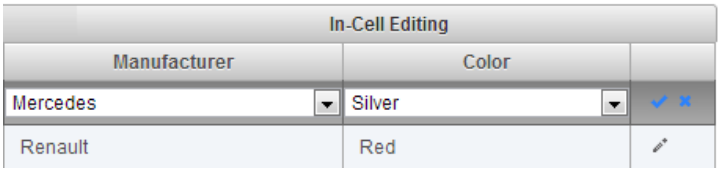


Figura 43. Arquitectura MVC

5.12 Exporter

El exporter és un element que ens permet exportar a arxius vistes de l'aplicació. Començant per la part visual, es renderitzen uns botons que estan associats a una vista (per exemple la taula del apartat anterior). Al clicar en un d'ells es crida la lògica .java associada indicant via paràmetre d'event quin format s'ha escollit. A la part servidor, es crea un canal basat en la coneguda classe OuputStream i depenen el format s'utilitza una llibreria o una altre *figura 44*.

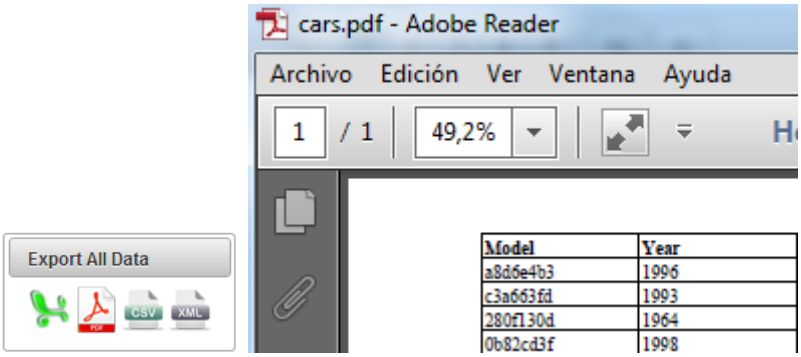


Figura 44. Arquitectura MVC

6. Proves, Manteniment i actualitzacions

6.1 Introducció

Per tal de generar un software de qualitat, hem de testejar-lo, mantenir-lo i actualitzar-lo. Per això en aquest capítol tractem algunes propostes per testejar de manera unitària així com seguir el cicle de vida del framework en mode debug.

Per altra banda donem resposta a com fer còpies de seguretat i actualitzacions o ampliacions.

6.2 Unit Testing

Una de les pràctiques habituals per testejar i validar codi és el test unitari. Consisteix en escriure mètodes per examinar les funcions de l'aplicació per separat. El mètode simula la crida a una funció de la nostra aplicació i verifica si el resultat éss l'espera't.

No obstant amb JSF no és tan fàcil dur a terme aquesta pràctica. En la majoria de cassos utilitzem la classe FacesContext (conté informació del entorn exterior al bean on l'utilitzem ja que aquest no pot accedir-hi). Quan es fa un request el servlet controlador crea un context (queda executant-se en un thread) que pot ser referenciat des de llavors via el mètode getCurrentInstance().

Els test unitaris es fan tenint en compte nomes la funció a provar, per tant en aquest cas al testeja un mètode d'un Bean que contingui una crida a FacesContext ens generarà un error per falta del context.

La solució per poder testejar l'hem copiat del líder de Primefaces (Optimus Prime) de la seva entrada de bloc: "Easy Unit Testing JSF Backing Beans" i consisteix en:

1. Crear una nova interficie FacesMessages amb metodes simples:
    a. Public void addError(String summary, String detail);
2. Pasar per parametre una instancia de prova del nou FacesMessages al setFacesMessages.

3. Al Bean substituir el missatge d'error per un de nou generat amb el nou Faces Messages.
4. Fer els asserts sobre l'instancia de prova.

6.3 JSF cicle debug

El framework JSF té un cicle de vida bastant difícil de seguir degut a la combinació de pàgines client i pàgines servidor. No obstant, segueix sis fases reconegudes com a:

- Restore view: es crea un arbre de components el primer cop que s'executa l'aplicació i els següents cops es restaura directament el arbre.
- Apply Request Values: s'itera els component del arbre per buscar el que concorda amb el valor de petició.
- Process Validations: realització de les validacions i conversions.
- Update Model Values: es modifiquen els valors associats a la vista que comuniquen amb els beans.
- Invoke Application: s'invoca el mètode associat als valors del punt anterior.
- Render Response: el servidor retorna la pagina de resposta al navegador .

Per saber en que fase es troba l'app (en el cas de no saber de quin punt ve un error per exemple) tenim les classes:

```
import javax.faces.event.PhaseEvent;
import javax.faces.event.PhaseId;
import javax.faces.event.PhaseListener;


public void beforePhase(PhaseEvent event) {
    System.out.println("Fase Anterior: " + event.getPhaseId());
          }
```

Fent print o debug amb aquest mètodes sabem en quina fase ens trobem determinades per phaseID i el nom.

6.4 Bbdd backup

En aquest apartat expliquem com fer una còpia de seguretat de les dades de l'aplicació (emmagatzemades a la bbdd).

Quan parlem de backups sobre Oracle tenim dues possibilitats: la còpia en fred o en calent. Mentre que la còpia en fred necessita que aturem el motor, la còpia en calent ho fa sense haver d'interrompre el servei. El cas fred té associat la tasca de copiar uns arxius de manera manual mentre que el calent sol ser la crida d'una funció mitjançant una consola que amb fitxers temporals va generant un document amb les dades.

La base de dades que utilitzem (Oracle 11g) ens proporciona una utilitat anomenada Data Pump Export. Al utilitzar-la, bolquem dades a un fitxer amb extensió .dmp. Dins de la mateixa podem exportarà esquemes, taules i/o tablespace a la vegada que també podem exportar-ho tot. És una còpia en calent i no necessita aturar la bbdd.

Per automatitzar la tasca, en un horari que no s'utilitzi l'aplicació (per evitar moviments, ja que al ser calent es podria fer en qualsevol moment). Fem ús de les tasques programades de Windows:

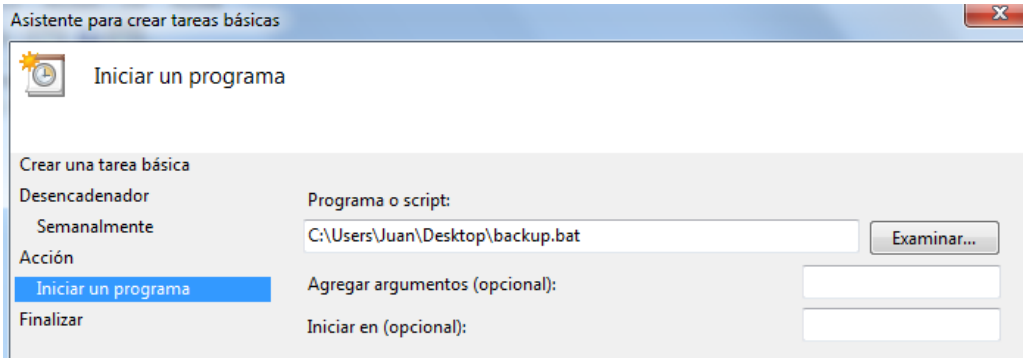


Figura 45. Tasca programada de Windows

Com podem observar, la tasca executarà setmanalment el arxiu backup.bat que conté:

```
sqlplus /nolog @backup.sql
expdp PFCAdmin/admin dumpfile=bak.dmp directory=EX_FOLDER full=y
```

I backup.sql que conté:

```
connect PFCAdmin/admin CREATE DIRECTORY EX_FOLDER AS
'd:/backup';
```

D'aquesta manera a creem un arxiu.dmp que en cas de error en el sistema o qualsevol incidència podem carregar de manera anàloga:

```
impdp PFCAdmin/admin directory 'd:/backup' dumpfile= bak.dmp
logfile=imp.log full=y;
```

6.5 Uprading

Degut a haver dissenyat i implementat l'aplicació de manera modular, amb un acoblament baix i per packages, el ampliar les funcionalitats de la mateixa és una tasca amb una seqüència molt definida i que no afecta a les altres funcionalitats. A més a més el client, degut a que es via web, no patirà problemes d'instal·lació o actualització.

A continuació detallem els passos d'un possible escenari d'upgrading:

1. Degut a que el client vol un nou mòdul de galeria de fotos es recullen requisits i es genera un document de disseny (seguint la metodologia emprada anteriorment al capítol Disseny).
2. Es crea un servidor de desenvolupament.
3. Un programador front-end amb l'ajuda d'un maquetador CSS generen una vista XHTML.
4. Un programador back-end implementa la lògica del sistema i les estructures de dades en arxius .java dins d'un package.
5. El client veu mitjançant una URL de pre-producció els resultats i accepta (després de les iteracions habituals i correccions).
6. Al servidor de producció es puja el package, la vista i es fa deploy de les mateixes.
7. L'usuari al accedir a la URL veu els canvis.

7. Ampliacions futures

7.1 Introducció

Si bé el conjunt de resultats obtinguts en aquest projecte satisfan inicialment el objectiu plantejat, encara considerem que hi ha una part que hauria de formar part per completar-ho. Degut a la limitació temporal, no s'ha pogut dur a terme però s'ha estudiat la seva viabilitat, i s'ha plantejat la línea a seguir per implementar-ho en el futur.

Per una part, tot i que l'aplicació té comunicació efectiva amb la bbdd, a l'hora de transformar aquest projecte en un producte comercial, necessitarem una capa de persistència de base de dades més potent que permeti mapejar el model relacional amb els objectes Java.

L'altre punt és el capacitar amb la tecnologia adient el projecte per tal de que es permeti la comunicació amb dispositius mòbils. Un exemple seria, enviar un missatge de text al UsuariManteniment quan té sense gestionar x tasques.

7.2 Ampliacions

7.2.1 Hibernate

Hibernate és l'eina que hem decidit utilitzar en el futur per a la capa de persistència. Aquesta, proporciona elements per a facilitar el mapeig entre la base de dades i els objectes de l'aplicació mitjançant arxius que estableixen les relacions. D'aquesta manera dotaríem tota l'aplicació amb la visió orientada a objectes i no hauríem de gastar temps en les conversions com ara.

A continuació posem un petit exemple per aclarir l'idea:

	ID	NOM
1	147	Linea1
2	148	Linea2
3	149	Linea3
4	150	Linea4
5	151	Linea5

Figura 46. Taula linea de la bbdd

Tenim aquesta taula i un package Jdbd per fer les comunicacions. Actualment, amb llenguatge SQL hem de realitzar unes 10 línees de codi per fer el SELECT, INSERT,... més una funció que iteri els resultats i que els vagi guardant en objectes java.

El que es pretén amb Hibernate es treballar sobre objectes Java i ho faríem:

```
public class Linea {

    private String id;
    private String nom;

        ...}
```

```
<hibernate-mapping>

    <class name="hibernate.Linea" table="LINEA">

        <id name="id" type="string">
           <column name="LINEA_ID" not-null="true"/>
        </id>

        <property name="name">
            <column name="NAME" not-null="true"/>
        </property>

    </class>

          </hibernate-mapping>
```

```
Linea line = new Linea();

line.setID(12356);
line.setSex('Linea1');
```

7.2.2 Android

Actualment els dispositius mòbils com ara smartphones o tablets disposen de la capacitat suficient per interactuar amb la web i les aplicacions que alberguen. Per aquest motiu hem previst que, si enfoquem ClientLight com a producte, ha de satisfer els futurs requisits referents a aquestes tecnologies.

Un Servlet al servidor ofereix un servei:

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response {
                         out.println("Hello Android !!!!");
```

Creem una activitat Android per a cridar el Servlet:

```
<activity android:name=".HttpGetServletActivity" .../>
```

Al arxiu de l'activitat principalment creem la URL, la connectem i fem getInputStream() per obtenir el resultat del Servlet:

```
1) static String URL ="http://url/HttpGetServlet/Servlet";

2) private InputStream getHttpConnection(String urlString)

3) try {
     HttpURLConnection httpConnection = (HttpURLConnection)
     url.openConnection();
     httpConnection.connect();
     stream = httpConnection.getInputStream();
```

7.3 Don't use jQuery to write Web Applications

Gairebé finalitzant el projecte, llegint articles sobre jQuery va aparèixer un article que hem cregut convenient per a compartir la reflexió que extraiem. El blog s'anomena Foy's blog de Julien Richard i l'article: "Don't use jQuery to write a web application". En general fa tres entrades al blog on destaca:

1. Un cop jQuery ha enmagatzemat els selectors que manipulem, cada cop que els requerim la cerca a través del arbre que utilitza és molt lenta gastant molt processador.
2. Al utilitzar jQuery tots els objectes s'envolten per un objecte jQuery que no permet utilitzar les API del paradigma DocumentObjectModel natiu.
3. No és òptim per a escriure aplicacions de gran grandària degut a que no considera que sigui la millor opció per al parsing de markup (com xml), gestionar estat d'objectes o comunicar-se amb la part lògica.

Per l'experiència que ens ha donat el projecte podem dir que, certament, els problemes i on hem gastat més temps és en els problemes citats. No obstant no coneixem una altra tecnologia i per això no es pot donar un veredicte consistent.

Ara bé, hem cercat per sobre les possibles solucions que hi ha a la web per en un futur poder confrontar el ClientLight fet amb JSF, jQuery i Primefaces amb un altre.

Creiem que una bona aposta seria una eina que tingués una dinàmica com la llibreria Swing de Java. Per això, seguint les tendències actuals, en un futur es pretén muntar ClientLight amb el toolkit GoogleWebToolkit (GWT). L'idea és com hem anomenat, treballar només sobre Java i amb mires d'aplicacions extenses. Els criteris per escollir-lo ha estat: disponibles plugins i frameworks per a JBoss i Eclipse, gran quantitat de documentació i de qualitat, freeware ,comunitat de suport ben consolidada i crítiques molt favorables a la xarxa.

8. Conclusions

8.1.1 Desviacions

Com es pot observar al capítol de planificació, l'entrega estava prevista per a Juliol. Es evident que no s'ha arribat degut a:

- Un augment en la meva jornada laboral escurçant el temps per a dedicar sobre la planificació prevista.
- Hem faltava temps per acabar d'investigar la llista de tecnologies, paradigmes i idees que volia assegurar que tindrien cabuda al producte.
- De la llista de funcionalitats, no totes han estat implementades degut al canvi d'orientació del projecte de funcional a creació d'un producte.

8.1.2 Valoracions personals

Un cop finalitzat, he de dir que estic molt satisfet del ClientLight. Per començar, quan encara no tenia nom, pretenia ser un portal amb algunes funcionalitats que resolguessin algunes qüestions d'organització empresarial. Poc a poc vaig anar recopilant informació, tenint un munt d'idees que de sobte un dia es van canalitzar en no centrar-me en fer quatre pantalles funcionals si no crear els ciments d'un producte.

Entenc que a vegades el tema del projecte de final de carrera està pensat per poder passar-ho de manera segura però aquest és un tema arriscat i l'inici d'una línea de treball.

Si ben es veritat que no s'han implementat totes les funcionalitats previstes, el meu veredicte és que darrera de les investigacions i decisions hi ha una combinació de factors que fan que siguin uns ciments molt bons per a la construcció de qualsevol producte.

El disseny és especial per a ClientLight però a l'hora el pot entendre qualsevol programador ja que està basat en UML.

L'implementació és molt sencilla i eficaç a la vegada que ofereix uns resultats amb bon rendiment i un dinamisme elevat. El codi és molt estàndard i des de un

dissenyador, maquetador, programador back-end o programador front-end ho pot entendre.

A curt termini estaran les noves funcinalitats de persistència i comunicació amb dispositius mòbils fent el producte més interessant.

A falta de proves oficials, sembla que el rendiment podrà ser comparable amb les aplicacions d'escriptori.

9. Bibliografia i Webgrafia

NOTA: tot el material ha estat consultat digitalment i la última revisió de les 07/09/2013.

(Articles teòrics sobre actius)
www.wikipedia.org

(Institut Assets Management)
www.theiam.org

(Comparativa estat de l'art)
www.technologyevaluation.org

(EAM vs. ERP)
www.confiabilidad.net

(Database download i tutorials)
www.oracle.com

(Demo gratis d'una aplicació)
www.infor.com

(Llibres, articles, exemples, descàrrega de la llibreria)
www.primefaces.org

(Tutorial hibernate)
www.hibernate.org

(backup)
http://www.ajpdsoft.com

(Tutorial Android, JSON, dubtes en general)
www.stackoverflow.com

(Android Servlet)
www.theopentutorials.com

(Article "don't use jQuery")
www.richard-foy.fr

(Google Developers (JavaScript Gmaps, GWT))
www.developers.google.com

Manual JSF, Junta Andalucía 2010
Core Servlets and JSP, Java Editions 2008